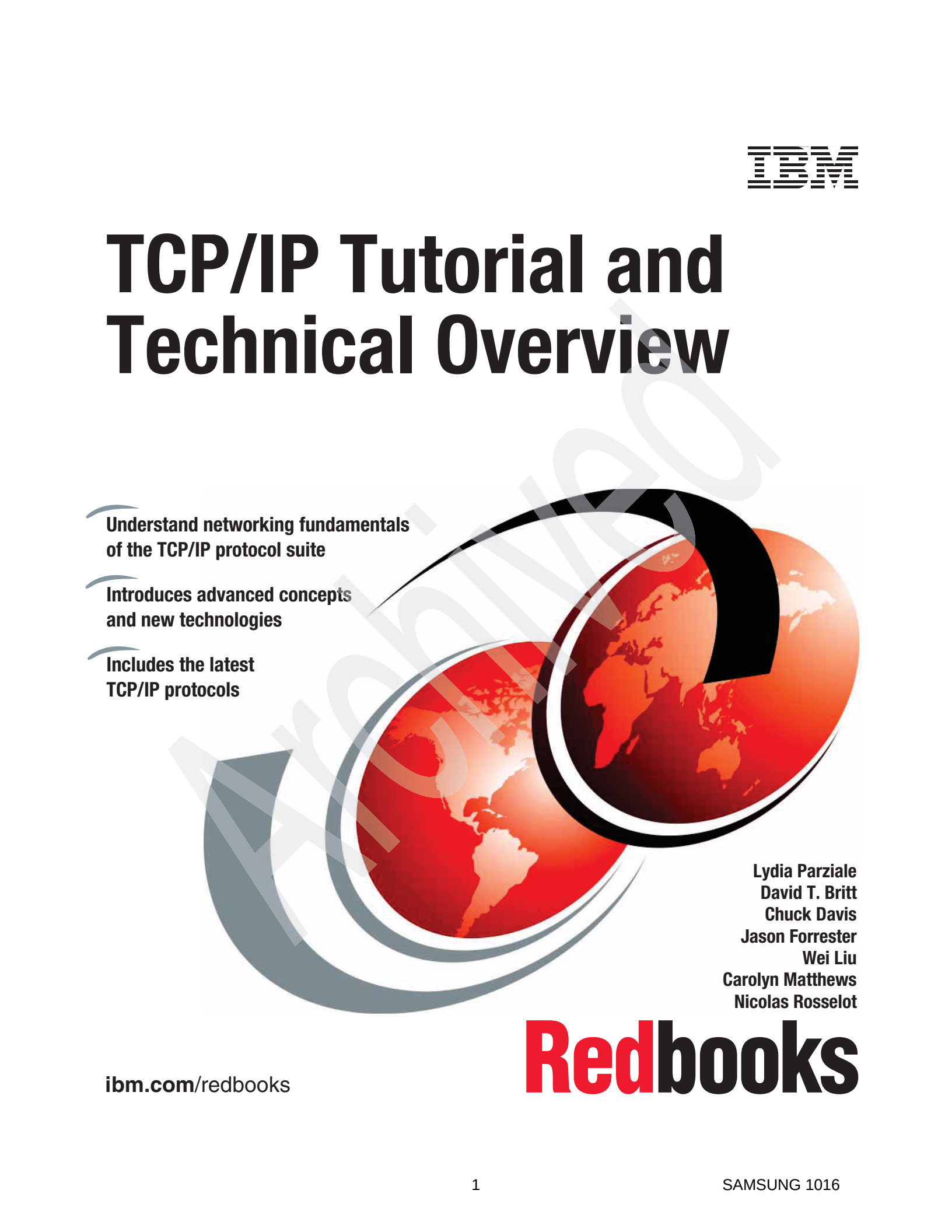


Archive

SAMSUNG 1016

Archived



International Technical Support Organization

TCP/IP Tutorial and Technical Overview

December 2006

Archived

GG24-3376-07

Note: Before using this information and the product it supports, read the information in "Notices" on page xvii.

Archived

Eighth Edition (December 2006)

© Copyright International Business Machines Corporation 1989-2006. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices xvii
Trademarks xviii

Preface xix
The team that wrote this redbook xx
Become a published author xxii
Comments welcome xxiii

Part 1. Core TCP/IP protocols 1

Chapter 1. Architecture, history, standards, and trends 3
1.1 TCP/IP architectural model 4
1.1.1 Internetworking 4
1.1.2 The TCP/IP protocol layers 6
1.1.3 TCP/IP applications 9
1.2 The roots of the Internet 12
1.2.1 ARPANET 14
1.2.2 NSFNET 15
1.2.3 Commercial use of the Internet 16
1.2.4 Internet2 18
1.2.5 The Open Systems Interconnection (OSI) Reference Model 20
1.3 TCP/IP standards 21
1.3.1 Request for Comments (RFC) 22
1.3.2 Internet standards 24
1.4 Future of the Internet 26
1.4.1 Multimedia applications 26
1.4.2 Commercial use 26
1.4.3 The wireless Internet 27
1.5 RFCs relevant to this chapter 27

Chapter 2. Network interfaces 29
2.1 Ethernet and IEEE 802 local area networks (LANs) 30
2.1.1 Gigabit Ethernet 33
2.2 Fiber Distributed Data Interface (FDDI) 33
2.3 Serial Line IP (SLIP) 34
2.4 Point-to-Point Protocol (PPP) 35
2.4.1 Point-to-point encapsulation 37
2.5 Integrated Services Digital Network (ISDN) 38
2.6 X.25 39

© Copyright IBM Corp. 1989-2006. All rights reserved.

Archived

2.7 Frame relay 41
2.7.1 Frame format 41
2.7.2 Interconnect issues 43
2.7.3 Data link layer parameter negotiation 43
2.7.4 IP over frame relay 44
2.8 PPP over SONET and SDH circuits 45
2.8.1 Physical layer 46
2.9 Multi-Path Channel+ (MPC+) 46
2.10 Asynchronous transfer mode (ATM) 47
2.10.1 Address resolution (ATMARP and InATMARP) 47
2.10.2 Classical IP over ATM 50
2.10.3 ATM LAN emulation 56
2.10.4 Classical IP over ATM versus LAN emulation 59
2.11 Multiprotocol over ATM (MPOA) 60
2.11.1 Benefits of MPOA 60
2.11.2 MPOA logical components 61
2.11.3 MPOA functional components 62
2.11.4 MPOA operation 63
2.12 RFCs relevant to this chapter 64

Chapter 3. Internetworking protocols 67
3.1 Internet Protocol (IP) 68
3.1.1 IP addressing 68
3.1.2 IP subnets 72
3.1.3 IP routing 77
3.1.4 Methods of delivery: Unicast, broadcast, multicast, and anycast . . . 84
3.1.5 The IP address exhaustion problem 86
3.1.6 Intranets: Private IP addresses 89
3.1.7 Network Address Translation (NAT) 89
3.1.8 Classless Inter-Domain Routing (CIDR) 95
3.1.9 IP datagram 98
3.2 Internet Control Message Protocol (ICMP) 109
3.2.1 ICMP messages 110
3.2.2 ICMP applications 117
3.3 Internet Group Management Protocol (IGMP) 119
3.4 Address Resolution Protocol (ARP) 119
3.4.1 ARP overview 119
3.4.2 ARP detailed concept 120
3.4.3 ARP and subnets 123
3.4.4 Proxy-ARP or transparent subnetting 123
3.5 Reverse Address Resolution Protocol (RARP) 124
3.5.1 RARP concept 125
3.6 Bootstrap Protocol (BOOTP) 125

3.6.1 BOOTP forwarding . . . 129
3.6.2 BOOTP considerations . . . 130
3.7 Dynamic Host Configuration Protocol (DHCP) . . . 130
3.7.1 The DHCP message format . . . 132
3.7.2 DHCP message types . . . 134
3.7.3 Allocating a new network address . . . 134
3.7.4 DHCP lease renewal process . . . 137
3.7.5 Reusing a previously allocated network address . . . 138
3.7.6 Configuration parameters repository . . . 139
3.7.7 DHCP considerations . . . 139
3.7.8 BOOTP and DHCP interoperability . . . 140
3.8 RFCs relevant to this chapter . . . 140

Chapter 4. Transport layer protocols . . . 143
4.1 Ports and sockets . . . 144
4.1.1 Ports . . . 144
4.1.2 Sockets . . . 145
4.2 User Datagram Protocol (UDP) . . . 146
4.2.1 UDP datagram format . . . 147
4.2.2 UDP application programming interface . . . 149
4.3 Transmission Control Protocol (TCP) . . . 149
4.3.1 TCP concept . . . 150
4.3.2 TCP application programming interface . . . 164
4.3.3 TCP congestion control algorithms . . . 165
4.4 RFCs relevant to this chapter . . . 170

Chapter 5. Routing protocols . . . 171
5.1 Autonomous systems . . . 173
5.2 Types of IP routing and IP routing algorithms . . . 174
5.2.1 Static routing . . . 175
5.2.2 Distance vector routing . . . 176
5.2.3 Link state routing . . . 177
5.2.4 Path vector routing . . . 178
5.2.5 Hybrid routing . . . 180
5.3 Routing Information Protocol (RIP) . . . 180
5.3.1 RIP packet types . . . 180
5.3.2 RIP packet format . . . 181
5.3.3 RIP modes of operation . . . 182
5.3.4 Calculating distance vectors . . . 182
5.3.5 Convergence and counting to infinity . . . 185
5.3.6 RIP limitations . . . 189
5.4 Routing Information Protocol Version 2 (RIP-2) . . . 189
5.4.1 RIP-2 packet format . . . 190

Archived

5.4.2 RIP-2 limitations 192
5.5 RIPng for IPv6 192
5.5.1 Differences between RIPng and RIP-2 193
5.5.2 RIPng packet format 193
5.6 Open Shortest Path First (OSPF) 196
5.6.1 OSPF terminology 196
5.6.2 Neighbor communication 205
5.6.3 OSPF neighbor state machine 206
5.6.4 OSPF route redistribution 208
5.6.5 OSPF stub areas 210
5.6.6 OSPF route summarization 211
5.7 Enhanced Interior Gateway Routing Protocol (EIGRP) 212
5.7.1 Features of EIGRP 212
5.7.2 EIGRP packet types 214
5.8 Exterior Gateway Protocol (EGP) 215
5.9 Border Gateway Protocol (BGP) 215
5.9.1 BGP concepts and terminology 216
5.9.2 IBGP and EBGP communication 218
5.9.3 Protocol description 220
5.9.4 Path selection 223
5.9.5 BGP synchronization 226
5.9.6 BGP aggregation 228
5.9.7 BGP confederations 230
5.9.8 BGP route reflectors 231
5.10 Routing protocol selection 233
5.11 Additional functions performed by the router 234
5.12 Routing processes in UNIX-based systems 235
5.13 RFCs relevant to this chapter 235

Chapter 6. IP multicast 237
6.1 Multicast addressing 238
6.1.1 Multicasting on a single physical network 238
6.1.2 Multicasting between network segments 240
6.2 Internet Group Management Protocol (IGMP) 241
6.2.1 IGMP messages 241
6.2.2 IGMP operation 247
6.3 Multicast delivery tree 250
6.4 Multicast forwarding algorithms 252
6.4.1 Reverse path forwarding algorithm 252
6.4.2 Center-based tree algorithm 253
6.4.3 Multicast routing protocols 254
6.5 Distance Vector Multicast Routing Protocol (DVMRP) 254
6.5.1 Protocol overview 254

6.5.2 Building and maintaining multicast delivery trees . . . 256
6.5.3 DVMRP tunnels . . . 258
6.6 Multicast OSPF (MOSPF) . . . 258
6.6.1 Protocol overview . . . 259
6.6.2 MOSPF and multiple OSPF areas . . . 260
6.6.3 MOSPF and multiple autonomous systems . . . 260
6.6.4 MOSPF interoperability . . . 261
6.7 Protocol Independent Multicast (PIM) . . . 261
6.7.1 PIM dense mode . . . 262
6.7.2 PIM sparse mode . . . 263
6.8 Interconnecting multicast domains . . . 266
6.8.1 Multicast Source Discovery Protocol (MSDP) . . . 266
6.8.2 Border Gateway Multicast Protocol . . . 269
6.9 The multicast backbone . . . 269
6.9.1 MBONE routing . . . 270
6.9.2 Multicast applications . . . 271
6.10 RFCs relevant to this chapter . . . 272

Chapter 7. Mobile IP . . . 275
7.1 Mobile IP overview . . . 276
7.1.1 Mobile IP operation . . . 277
7.1.2 Mobility agent advertisement extensions . . . 278
7.2 Mobile IP registration process . . . 280
7.2.1 Tunneling . . . 284
7.2.2 Broadcast datagrams . . . 284
7.2.3 Move detection . . . 284
7.2.4 Returning home . . . 285
7.2.5 ARP considerations . . . 285
7.2.6 Mobile IP security considerations . . . 286
7.3 RFCs relevant to this chapter . . . 286

Chapter 8. Quality of service . . . 287
8.1 Why QoS? . . . 288
8.2 Integrated Services . . . 289
8.2.1 Service classes . . . 292
8.2.2 Controlled Load Service . . . 294
8.2.3 Guaranteed Service . . . 295
8.2.4 The Resource Reservation Protocol (RSVP) . . . 296
8.2.5 Integrated Services outlook . . . 308
8.3 Differentiated Services . . . 309
8.3.1 Differentiated Services architecture . . . 310
8.3.2 Organization of the DSCP . . . 313
8.3.3 Configuration and administration of DS with LDAP . . . 322

8.4 RFCs relevant to this chapter 325

Chapter 9. IP version 6 327
9.1 IPv6 introduction 328
9.1.1 IP growth 328
9.1.2 IPv6 feature overview 330
9.2 The IPv6 header format 330
9.2.1 Extension headers 333
9.2.2 IPv6 addressing 339
9.2.3 Traffic class 345
9.2.4 Flow labels 346
9.2.5 IPv6 security 347
9.2.6 Packet sizes 350
9.3 Internet Control Message Protocol Version 6 (ICMPv6) 352
9.3.1 Neighbor discovery 353
9.3.2 Multicast Listener Discovery (MLD) 365
9.4 DNS in IPv6 367
9.4.1 Format of IPv6 resource records 368
9.5 DHCP in IPv6 371
9.5.1 DHCPv6 messages 371
9.6 IPv6 mobility support 372
9.7 IPv6 new opportunities 376
9.7.1 New infrastructure 376
9.7.2 New services 377
9.7.3 New research and development platforms 378
9.8 Internet transition: Migrating from IPv4 to IPv6 379
9.8.1 Dual IP stack implementation: The IPv6/IPv4 node 380
9.8.2 Tunneling 381
9.8.3 Interoperability summary 388
9.9 RFCs relevant to this chapter 389

Chapter 10. Wireless IP 391
10.1 Wireless concepts 392
10.2 Why wireless? 395
10.2.1 Deployment and cost effectiveness 395
10.2.2 Reachability 396
10.2.3 Scalability 396
10.2.4 Security 397
10.2.5 Connectivity and reliability 397
10.3 WiFi 397
10.4 WiMax 400
10.5 Applications of wireless networking 402
10.5.1 Last mile connectivity in broadband services 402

10.5.2 Hotspots . . . 402
10.5.3 Mesh networking . . . 402
10.6 IEEE standards relevant to this chapter . . . 403

Part 2. TCP/IP application protocols . . . 405

Chapter 11. Application structure and programming interfaces . . . 407
11.1 Characteristics of applications . . . 408
11.1.1 The client/server model . . . 408
11.2 Application programming interfaces (APIs) . . . 410
11.2.1 The socket API . . . 410
11.2.2 Remote Procedure Call (RPC) . . . 415
11.2.3 The SNMP distributed programming interface (SNMP DPI) . . . 419
11.2.4 REXX sockets . . . 422
11.3 RFCs relevant to this chapter . . . 423

Chapter 12. Directory and naming protocols . . . 425
12.1 Domain Name System (DNS) . . . 426
12.1.1 The hierarchical namespace . . . 426
12.1.2 Fully qualified domain names (FQDNs) . . . 428
12.1.3 Generic domains . . . 428
12.1.4 Country domains . . . 429
12.1.5 Mapping domain names to IP addresses . . . 429
12.1.6 Mapping IP addresses to domain names: Pointer queries . . . 430
12.1.7 The distributed name space . . . 430
12.1.8 Domain name resolution . . . 432
12.1.9 Domain Name System resource records . . . 436
12.1.10 Domain Name System messages . . . 439
12.1.11 A simple scenario . . . 445
12.1.12 Extended scenario . . . 449
12.1.13 Transport . . . 450
12.1.14 DNS applications . . . 451
12.2 Dynamic Domain Name System . . . 453
12.2.1 Dynamic updates in the DDNS . . . 454
12.2.2 Incremental zone transfers in DDNS . . . 456
12.2.3 Prompt notification of zone transfer . . . 457
12.3 Network Information System (NIS) . . . 458
12.4 Lightweight Directory Access Protocol (LDAP) . . . 459
12.4.1 LDAP: Lightweight access to X.500 . . . 460
12.4.2 The LDAP directory server . . . 461
12.4.3 Overview of LDAP architecture . . . 463
12.4.4 LDAP models . . . 464
12.4.5 LDAP security . . . 471
12.4.6 LDAP URLs . . . 474

12.4.7 LDAP and DCE 475
12.4.8 The Directory-Enabled Networks (DEN) initiative 477
12.4.9 Web-Based Enterprise Management (WBEM) 478
12.5 RFCs relevant to this chapter 478

Chapter 13. Remote execution and distributed computing 483
13.1 Telnet 484
13.1.1 Telnet operation 484
13.1.2 Network Virtual Terminal 485
13.1.3 Telnet options 487
13.1.4 Telnet command structure 489
13.1.5 Option negotiation 491
13.1.6 Telnet basic commands 492
13.1.7 Terminal emulation (Telnet 3270) 492
13.1.8 TN3270 enhancements (TN3270E) 493
13.1.9 Device-type negotiation 494
13.2 Remote Execution Command protocol (REXEC and RSH) 495
13.3 Introduction to the Distributed Computing Environment (DCE) 496
13.3.1 DCE directory service 498
13.3.2 Authentication service 502
13.3.3 DCE threads 505
13.3.4 Distributed Time Service 507
13.3.5 Additional information 509
13.4 Distributed File Service (DFS) 509
13.4.1 File naming 510
13.4.2 DFS performance 511
13.5 RFCs relevant to this chapter 512

Chapter 14. File-related protocols 513
14.1 File Transfer Protocol (FTP) 514
14.1.1 An overview of FTP 514
14.1.2 FTP operations 515
14.1.3 The active data transfer 520
14.1.4 The passive data transfer 521
14.1.5 Using proxy transfer 522
14.1.6 Reply codes 523
14.1.7 Anonymous FTP 525
14.1.8 Using FTP with IPv6 525
14.1.9 Securing FTP sessions 527
14.2 Trivial File Transfer Protocol (TFTP) 529
14.2.1 TFTP usage 530
14.2.2 Protocol description 531
14.2.3 TFTP packets 531

Archived

14.2.4 Data modes . . . 532
14.2.5 TFTP multicast option . . . 532
14.2.6 Security issues . . . 533
14.3 Secure Copy Protocol (SCP) and SSH FTP (SFTP) . . . 533
14.3.1 SCP syntax and usage . . . 533
14.3.2 SFTP syntax and usage . . . 535
14.3.3 SFTP interactive commands . . . 536
14.4 Network File System (NFS) . . . 538
14.4.1 NFS concept . . . 538
14.4.2 File integrity . . . 542
14.4.3 Lock Manager protocol . . . 543
14.4.4 NFS file system . . . 543
14.4.5 NFS version 4 . . . 543
14.4.6 Cache File System . . . 545
14.4.7 WebNFS . . . 545
14.5 The Andrew File System (AFS) . . . 546
14.6 Common Internet File System (CIFS) . . . 548
14.6.1 NetBIOS over TCP/IP . . . 548
14.6.2 SMB/CIFS specifics . . . 550
14.7 RFCs relevant to this chapter . . . 552

Chapter 15. Mail applications . . . 555
15.1 Simple Mail Transfer Protocol . . . 556
15.1.1 How SMTP works . . . 559
15.1.2 SMTP and the Domain Name System . . . 565
15.2 Sendmail . . . 568
15.2.1 Sendmail as a mail transfer agent (MTA) . . . 568
15.2.2 How sendmail works . . . 569
15.3 Multipurpose Internet Mail Extensions (MIME) . . . 571
15.3.1 How MIME works . . . 574
15.3.2 The Content-Transfer-Encoding field . . . 582
15.3.3 Using non-ASCII characters in message headers . . . 587
15.4 Post Office Protocol (POP) . . . 589
15.4.1 Connection states . . . 589
15.4.2 POP3 commands and responses . . . 590
15.5 Internet Message Access Protocol (IMAP4) . . . 591
15.5.1 Fundamental IMAP4 electronic mail models . . . 591
15.5.2 IMAP4 states . . . 592
15.5.3 IMAP4 commands and response interaction . . . 594
15.5.4 IMAP4 messages . . . 597
15.6 RFCs relevant to this chapter . . . 599

Chapter 16. The Web . . . 601

16.1 Web browsers 603
16.2 Web servers 604
16.3 Hypertext Transfer Protocol (HTTP) 605
16.3.1 Overview of HTTP 605
16.3.2 HTTP operation 606
16.4 Content 615
16.4.1 Static content 615
16.4.2 Client-side dynamic content 616
16.4.3 Server-side dynamic content 617
16.4.4 Developing content with IBM Web application servers 621
16.5 RFCs relevant to this chapter 621

Chapter 17. Network management 623
17.1 The Simple Network Management Protocol (SNMP) 624
17.1.1 The Management Information Base (MIB) 625
17.1.2 The SNMP agent 630
17.1.3 The SNMP manager 631
17.1.4 The SNMP subagent 632
17.1.5 The SNMP model 633
17.1.6 SNMP traps 638
17.1.7 SNMP versions 639
17.1.8 Single authentication and privacy protocol 647
17.2 The NETSTAT utility 648
17.2.1 Common NETSTAT options 649
17.2.2 Sample NETSTAT report output 649
17.3 RFCs relevant to this chapter 651

Chapter 18. Wireless Application Protocol 655
18.1 The WAP environment 657
18.2 Key elements of the WAP specifications 657
18.3 WAP architecture 658
18.4 Client identifiers 663
18.5 Multimedia messaging system (MMS) 663
18.6 WAP push architecture 664
18.6.1 Push framework 664
18.6.2 Push proxy gateway (PPG) 665
18.6.3 Push access control protocol (PAP) 667
18.6.4 Service indication 668
18.6.5 Push over-the-air protocol (OTA) 668
18.6.6 Client-side infrastructure 668
18.6.7 Security 669
18.7 The Wireless Application Environment (WAE2) 670
18.8 User Agent Profile (UAProf) 671

18.9 Wireless protocols . . . 672
18.9.1 Wireless Datagram Protocol (WDP) . . . 672
18.9.2 Wireless Profiled Transmission Control Protocol (WP-TCP) . . . 674
18.9.3 Wireless Control Message Protocol (WCMP) . . . 678
18.9.4 Wireless Transaction Protocol (WTP) . . . 679
18.9.5 Wireless Session Protocol (WSP) . . . 682
18.9.6 Wireless profiled HTTP (W-HTTP) . . . 695
18.10 Wireless security . . . 696
18.10.1 Wireless Transport Layer Security (WTLS) . . . 696
18.10.2 Wireless Identity Module (WIM) . . . 701
18.11 Wireless Telephony Application (WTA) . . . 702
18.12 RFCs relevant to this chapter . . . 702
18.13 Specifications relevant to this chapter . . . 703

Chapter 19. Presence over IP . . . 707
19.1 Overview of the presence service . . . 710
19.2 Presence Information Data Format (PIDF) . . . 714
19.3 Presence protocols . . . 716
19.3.1 Binding to TCP . . . 718
19.3.2 Address resolution . . . 718
19.4 RFCs relevant to this chapter . . . 718

Part 3. Advanced concepts and new technologies . . . 721

Chapter 20. Voice over Internet Protocol . . . 723
20.1 Voice over IP (VoIP) introduction . . . 724
20.1.1 Benefits and applications . . . 724
20.1.2 VoIP functional components . . . 726
20.2 Session Initiation Protocol (SIP) technologies . . . 730
20.2.1 SIP request and response . . . 732
20.2.2 Sample SIP message flow . . . 733
20.2.3 SIP protocol architecture . . . 734
20.3 Media Gateway Control Protocol (MGCP) . . . 736
20.3.1 MGCP architecture . . . 737
20.3.2 MGCP primitives . . . 737
20.4 Media Gateway Controller (Megaco) . . . 738
20.4.1 Megaco architecture . . . 738
20.5 ITU-T recommendation H.323 . . . 739
20.5.1 H.323 architecture . . . 739
20.5.2 H.323 protocol stack . . . 741
20.6 Summary of VoIP protocols . . . 742
20.7 RFCs relevant to this chapter . . . 743

Chapter 21. Internet Protocol Television . . . 745

21.1 IPTV overview 746
21.1.1 IPTV requirements 747
21.1.2 Business benefits and applications 749
21.2 Functional components 750
21.2.1 Content acquisition 750
21.2.2 CODEC (encode and decode) 750
21.2.3 Display devices and control gateway 751
21.2.4 IP (TV) transport 752
21.3 IPTV technologies 752
21.3.1 Summary of protocol standards 753
21.3.2 Stream Control Transmission Protocol 753
21.3.3 Session Description Protocol 754
21.3.4 Real-Time Transport Protocol (RTP) 756
21.3.5 Real-Time Control Protocol 762
21.3.6 Moving Picture Experts Group (MPEG) standards 767
21.3.7 H.261 769
21.4 RFCs relevant to this chapter 770

Chapter 22. TCP/IP security 771
22.1 Security exposures and solutions 772
22.1.1 Common attacks against security 772
22.1.2 Solutions to network security problems 772
22.1.3 Implementations of security solutions 774
22.1.4 Network security policy 776
22.2 A short introduction to cryptography 777
22.2.1 Terminology 777
22.2.2 Symmetric or secret-key algorithms 779
22.2.3 Asymmetric or public key algorithms 780
22.2.4 Hash functions 785
22.2.5 Digital certificates and certification authorities 791
22.2.6 Random-number generators 792
22.2.7 Export/import restrictions on cryptography 793
22.3 Firewalls 794
22.3.1 Firewall concept 795
22.3.2 Components of a firewall system 796
22.3.3 Types of firewalls 805
22.4 IP Security Architecture (IPSec) 809
22.4.1 Concepts 810
22.4.2 Authentication Header (AH) 813
22.4.3 Encapsulating Security Payload (ESP) 817
22.4.4 Combining IPSec protocols 823
22.4.5 Internet Key Exchange (IKE) protocol 829
22.5 SOCKS 846

22.5.1 SOCKS Version 5 (SOCKSv5) 848
22.6 Secure Shell (1 and 2) 853
22.6.1 SSH overview 853
22.7 Secure Sockets Layer (SSL) 854
22.7.1 SSL overview 854
22.7.2 SSL protocol 856
22.8 Transport Layer Security (TLS) 861
22.9 Secure Multipurpose Internet Mail Extension (S-MIME) 861
22.10 Virtual private networks (VPNs) overview 861
22.10.1 VPN introduction and benefits 862
22.11 Kerberos authentication and authorization system 864
22.11.1 Assumptions 865
22.11.2 Naming 865
22.11.3 Kerberos authentication process 866
22.11.4 Kerberos database management 870
22.11.5 Kerberos Authorization Model 871
22.11.6 Kerberos Version 5 enhancements 871
22.12 Remote access authentication protocols 872
22.13 Extensible Authentication Protocol (EAP) 874
22.14 Layer 2 Tunneling Protocol (L2TP) 875
22.14.1 Terminology 876
22.14.2 Protocol overview 877
22.14.3 L2TP security issues 879
22.15 Secure Electronic Transaction (SET) 880
22.15.1 SET roles 880
22.15.2 SET transactions 881
22.15.3 The SET certificate scheme 883
22.16 RFCs relevant to this chapter 885

Chapter 23. Port based network access control 889
23.1 Port based network access control (NAC) overview 890
23.2 Port based NAC component overview 891
23.3 Port based network access control operation 892
23.3.1 Port based network access control functional considerations 904
23.4 RFCs relevant to this chapter 906

Chapter 24. Availability, scalability, and load balancing 907
24.1 Availability 909
24.2 Scalability 909
24.3 Load balancing 910
24.4 Clustering 910
24.5 Virtualization 912
24.6 Virtual Router Redundancy Protocol (VRRP) 914

24.6.1 Introduction 914
24.6.2 VRRP definitions 916
24.6.3 VRRP overview 916
24.6.4 Sample configuration 918
24.6.5 VRRP packet format 919
24.7 Round-robin DNS 921
24.8 Alternative solutions to load balancing 921
24.8.1 Network Address Translation 922
24.8.2 Encapsulation 923
24.9 RFCs relevant to this chapter 924

Appendix A. Multiprotocol Label Switching 925
A.1 MPLS: An introduction 926
A.1.1 Conventional routing versus MPLS forwarding mode 926
A.1.2 Benefits 927
A.1.3 Terminology 929
A.2 MPLS network processing 932
A.2.1 Label swapping 932
A.2.2 Label switched path (LSP) 934
A.2.3 Label stack and label hierarchies 934
A.2.4 MPLS stacks in a BGP environment 936
A.2.5 Label distribution protocols 938
A.2.6 Stream merge 939
A.3 Emulating Ethernet over MPLS networks 939
A.4 Generalized Multiprotocol Label Switching (GMPLS) 941
A.4.1 Benefits 941
A.4.2 MPLS and GMPLS comparison in OTN environment 942
A.4.3 How does GMPLS work? 943
A.4.4 Link Management Protocol (LMP) 944
A.4.5 Signaling for route selection and path setup 947
A.4.6 GMPLS considerations 949
A.4.7 GMPLS examples 950
A.5 RFCs relevant to this chapter 952

Abbreviations and acronyms 953

Related publications 959
IBM Redbooks 959
Other publications 959
Online resources 959
How to get IBM Redbooks 961
Help from IBM 961

Index 963

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 1989-2006. All rights reserved.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

1350™	IBM Global Network®	OS/390®
Advanced Peer-to-Peer Networking®	ibm.com®	OS/400®
AIX 5L™	IBM®	RACF®
AIX®	IPDS™	Redbooks (logo) ™
AS/400®	Lotus Notes®	Redbooks™
CICS®	Lotus®	RISC System/6000®
developerWorks®	MVS™	System/390®
ESCON®	Notes®	VTAM®
HiperSockets™	Operating System/2®	WebSphere®
	OS/2®	z/OS®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

CacheFS, Enterprise JavaBeans, EJB, IPX, Java, Java Naming and Directory Interface, JavaBeans, JavaScript, JavaServer, JavaServer Pages, JavaSoft, JDBC, JDK, JSP, JVM, J2EE, ONC, Solaris, Sun, Sun Microsystems, WebNFS, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Internet Explorer, Microsoft, MSN, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Archived

Preface

The TCP/IP protocol suite has become a staple of today's international society and global economy. Continually evolving standards provide a wide and flexible foundation on which an entire infrastructure of applications are built. Through these we can seek entertainment, conduct business, make financial transactions, deliver services, and much, much more.

However, because TCP/IP continues to develop and grow in order to meet the changing needs of our communities, it might sometimes be hard to keep track of new functionality or identify new possibilities. For this reason, the *TCP/IP Tutorial and Technical Overview* provides not only an introduction to the TCP/IP protocol suite, but also serves as a reference for advanced users seeking to keep their TCP/IP skills aligned with current standards. It is our hope that both the novice and the expert will find useful information in this publication.

In Part I, you will find an introduction to the core concepts and history upon which TCP/IP is founded. Included is an introduction to the history of TCP/IP and an overview of its current architecture. We also provide detailed discussions about the protocols that comprise the suite, and how those protocols are most commonly implemented.

Part II expands on the information provided in Part I, providing general application concepts (such as file sharing) and specific application protocols within those concepts (such as the File Transfer Protocol, or FTP). Additionally, Part II discusses applications that might not be included in the standard TCP/IP suite but, because of their wide use throughout the Internet community, are considered de facto standards.

Finally, Part III addresses new concepts and advanced implementations within the TCP/IP architecture. Of particular note, Part III examines the convergence of many formerly disparate networks and services using IP technology. Conjointly, this section reviews potential dangers of this IP convergence and approaches the ever-growing standards used to secure and control access to networks and networked resources.

We purposely kept this book platform independent. However, we recognize that you might have a need to learn more about TCP/IP on various platforms, so the following Web sites might assist you in further researching this topic:

- TCP/IP and System z:

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

© Copyright IBM Corp. 1989-2006. All rights reserved.

- TCP/IP and System p:

  http://www.ibm.com/systems/p/library/index.html

- TCP/IP and System i:

  http://www.ibm.com/servers/eserver/iseries/tcpip/index.html

- TCP/IP and System x:

  http://www.ibm.com/servers/eserver/support/xseries/allproducts/installing.html

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.



Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for 23 years in various technology areas.



David T. Britt is a Software Engineer for IBM in Research Triangle Park, NC, working specifically with the z/OS® Communications Server product. He is a subject matter expert in the Simple Networking Management Protocol (SNMP) and File Transfer Protocol (FTP), and has written educational material for both in the form of IBM Technotes, Techdocs, and Webcasts. He holds a degree in Mathematical Sciences from the University of North Carolina in Chapel Hill, and is currently pursuing a master of science in Information Technology and Management from the University of North Carolina in Greensboro.



Chuck Davis is a Security Architect in the U.S. He has 12 years of experience in IT security field. He has worked at IBM for nine years. His areas of expertise include IT security and privacy. He has written extensively about UNIX/Linux® and Internet security.



Jason Forrester is an IT Architect for IBM Global Technology Services in Boulder, CO. He has more than 12 years of experience with network communications. Specializing in IT strategy and architecture, Jason has designed large-scale enterprise infrastructures. He holds a CCIE certification and his work has lead to multiple patents on advanced networking concepts.

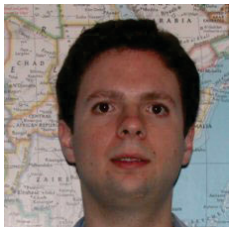


Dr. Wei Liu received his Ph.D. from Georgia Institute of Technology. He has taught TCP/IP networks in the University of Maryland (UMBC campus) and he has participated in ICCCN conference organization committees. Dr. Liu has given lectures at Sun™ Yat-Sen University and Shantou University in Next Generation Networks (NGNs). With more than 30 technical publications (in packet networks, telecommunications, and standards), he has received several awards from ATIS committees. Dr. Wei Liu has more than 10 years of telecom industry experience, having participated in various network transformation projects and service integration programs. Currently, he is investigating new infrastructure opportunities (virtualization, network, services, security, and metadata models) that can lead to future offering and new capabilities.

Archived



Carolyn Matthews is an IT Architect for IBM Global Technology Services in South Africa. She is an infrastructure architect for one of South Africa's largest accounts. She also acts as a consultant, using various IBM techniques. Carolyn holds an honors degree in Information Systems and is currently pursuing her master's degree in Information Systems. Her areas of expertise include TCP/IP networks, IT architecture, and new technologies.



Nicolas Rosselot is a Developer from Santiago, Chile. He has most recently been teaching an "Advanced TCP/IP Networking" class at Andres Bello University.

Thanks to the following people for their contributions to this project and laying the foundation for this book by writing the earlier version:

Adolfo Rodriguez, John Gatrell, John Karas, Roland Peschke, Srinath Karanam, and Martín F. Maldonado
International Technical Support Organization, Poughkeepsie Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM® Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  `ibm.com/redbooks`

- Send your comments in an e-mail to:

  `redbooks@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Archived

Archived

Part 1

Core TCP/IP protocols

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite has become the industry-standard method of interconnecting hosts, networks, and the Internet. As such, it is seen as the engine behind the Internet and networks worldwide.

Although TCP/IP supports a host of applications, both standard and nonstandard, these applications could not exist without the foundation of a set of core protocols. Additionally, in order to understand the capability of TCP/IP applications, an understanding of these core protocols must be realized.

With this in mind, Part I begins with providing a background of TCP/IP, the current architecture, standards, and most recent trends. Next, the section explores the two aspects vital to the IP stack itself. This portion begins with a discussion of the network interfaces most commonly used to allow the protocol suite to interface with the physical network media. This is followed by the protocols that must be implemented in any stack, including protocols belonging to the IP and transport layers.

© Copyright IBM Corp. 1989-2006. All rights reserved.

Finally, other standard protocols exist that might not necessarily be required in every implementation of the TCP/IP protocol suite. However, there are those that can be very useful given certain operational needs of the implementation. Such protocols include IP version 6, quality of service protocols, and wireless IP.

Archived

1

Architecture, history, standards, and trends

Today, the Internet and World Wide Web (WWW) are familiar terms to millions of people all over the world. Many people depend on applications enabled by the Internet, such as electronic mail and Web access. In addition, the increase in popularity of business applications places additional emphasis on the Internet. The Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is the engine for the Internet and networks worldwide. Its simplicity and power has led to its becoming the single network protocol of choice in the world today. In this chapter, we give an overview of the TCP/IP protocol suite. We discuss how the Internet was formed, how it developed, and how it is likely to develop in the future.

© Copyright IBM Corp. 1989-2006. All rights reserved.

1.1 TCP/IP architectural model

The TCP/IP protocol suite is so named for two of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP). A less used name for it is the Internet Protocol Suite, which is the phrase used in official Internet standards documents. In this book, we use the more common, shorter term, TCP/IP, to refer to the entire protocol suite.

1.1.1 Internetworking

The main design goal of TCP/IP was to build an interconnection of networks, referred to as an *internetwork*, or *internet*, that provided universal communication services over heterogeneous physical networks. The clear benefit of such an internetwork is the enabling of communication between hosts on different networks, perhaps separated by a large geographical area.

The words internetwork and internet are simply a contraction of the phrase interconnected network. However, when written with a capital “I”, the Internet refers to the worldwide set of interconnected networks. Therefore, the Internet is an internet, but the reverse does not apply. The Internet is sometimes called the *connected Internet*.

The Internet consists of the following groups of networks:

- Backbones: Large networks that exist primarily to interconnect other networks. Also known as network access points (NAPs) or Internet Exchange Points (IXPs). Currently, the backbones consist of commercial entities.
- Regional networks connecting, for example, universities and colleges.
- Commercial networks providing access to the backbones to subscribers, and networks owned by commercial organizations for internal use that also have connections to the Internet.
- Local networks, such as campus-wide university networks.

In most cases, networks are limited in size by the number of users that can belong to the network, by the maximum geographical distance that the network can span, or by the applicability of the network to certain environments. For example, an Ethernet network is inherently limited in terms of geographical size. Therefore, the ability to interconnect a large number of networks in some hierarchical and organized fashion enables the communication of any two hosts belonging to this internetwork.

Figure 1-1 shows two examples of internets. Each consists of two or more physical networks.

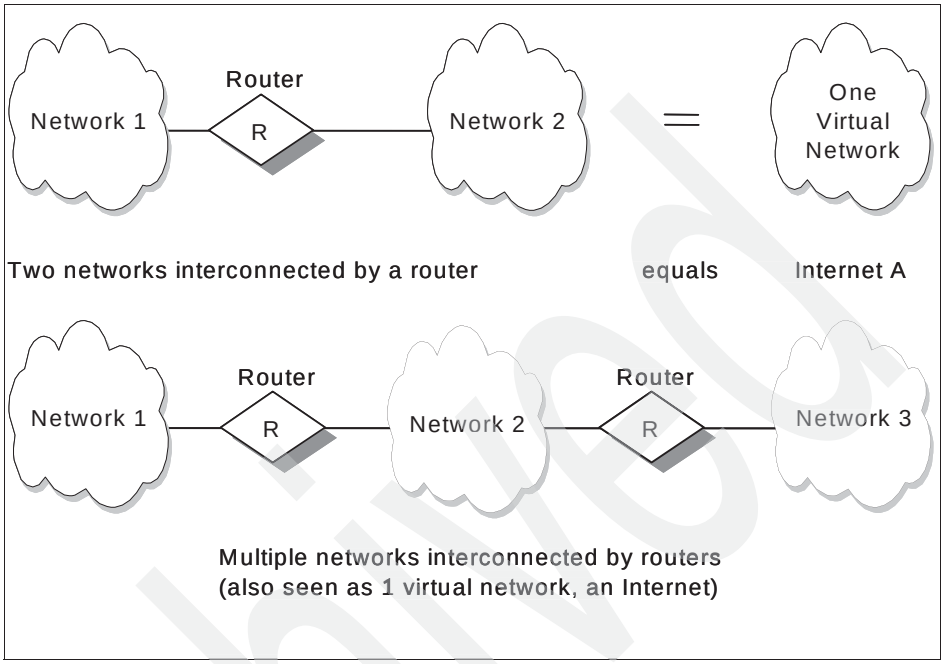


Figure 1-1 Internet examples: Two interconnected sets of networks, each seen as one logical network

Another important aspect of TCP/IP internetworking is the creation of a standardized abstraction of the communication mechanisms provided by each type of network. Each physical network has its own technology-dependent communication interface, in the form of a programming interface that provides basic communication functions (primitives). TCP/IP provides communication services that run between the programming interface of a physical network and user applications. It enables a common interface for these applications, independent of the underlying physical network. The architecture of the physical network is therefore hidden from the user and from the developer of the application. The application need only code to the standardized communication abstraction to be able to function under any type of physical network and operating platform.

As is evident in Figure 1-1, to be able to interconnect two networks, we need a computer that is attached to both networks and can forward data packets from one network to the other; such a machine is called a router. The term IP router is also used because the routing function is part of the Internet Protocol portion of the TCP/IP protocol suite (see 1.1.2, “The TCP/IP protocol layers” on page 6).

To be able to identify a host within the internetwork, each host is assigned an address, called the IP address. When a host has multiple network adapters (interfaces), such as with a router, each interface has a unique IP address. The IP address consists of two parts:

IP address = <network number><host number>

The network number part of the IP address identifies the network within the internet and is assigned by a central authority and is unique throughout the internet. The authority for assigning the host number part of the IP address resides with the organization that controls the network identified by the network number. We describe the addressing scheme in detail in 3.1.1, “IP addressing” on page 68.

1.1.2 The TCP/IP protocol layers

Like most networking software, TCP/IP is modeled in layers. This layered representation leads to the term protocol stack, which refers to the stack of layers in the protocol suite. It can be used for positioning (but not for functionally comparing) the TCP/IP protocol suite against others, such as Systems Network Architecture (SNA) and the Open System Interconnection (OSI) model. Functional comparisons cannot easily be extracted from this, because there are basic differences in the layered models used by the different protocol suites.

By dividing the communication software into layers, the protocol stack allows for division of labor, ease of implementation and code testing, and the ability to develop alternative layer implementations. Layers communicate with those above and below via concise interfaces. In this regard, a layer provides a service for the layer directly above it and makes use of services provided by the layer directly below it. For example, the IP layer provides the ability to transfer data from one host to another without any guarantee to reliable delivery or duplicate suppression. Transport protocols such as TCP make use of this service to provide applications with reliable, in-order, data stream delivery.

Figure 1-2 shows how the TCP/IP protocols are modeled in four layers.

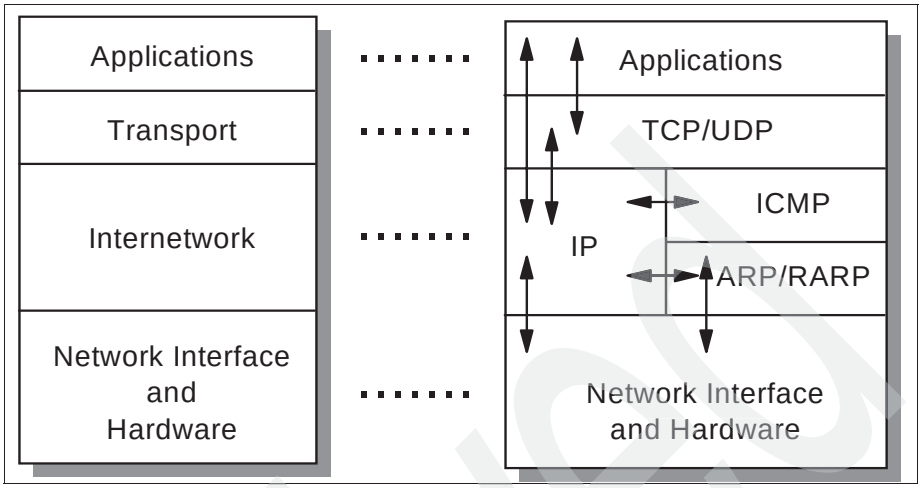


Figure 1-2 The TCP/IP protocol stack: Each layer represents a package of functions

These layers include:

Application layer The application layer is provided by the program that uses TCP/IP for communication. An application is a user process cooperating with another process usually on a different host (there is also a benefit to application communication within a single host). Examples of applications include Telnet and the File Transfer Protocol (FTP). The interface between the application and transport layers is defined by port numbers and sockets, which we describe in more detail in 4.1, “Ports and sockets” on page 144.

Transport layer The transport layer provides the end-to-end data transfer by delivering data from an application to its remote peer. Multiple applications can be supported simultaneously. The most-used transport layer protocol is the Transmission Control Protocol (TCP), which provides connection-oriented reliable data delivery, duplicate data suppression, congestion control, and flow control. We discuss this in more detail in 4.3, “Transmission Control Protocol (TCP)” on page 149.

Another transport layer protocol is the User Datagram Protocol (see 4.2, “User Datagram Protocol (UDP)” on page 146). It provides connectionless, unreliable,

best-effort service. As a result, applications using UDP as the transport protocol have to provide their own end-to-end integrity, flow control, and congestion control, if desired. Usually, UDP is used by applications that need a fast transport mechanism and can tolerate the loss of some data.

Internetwork layer

The internetwork layer, also called the *internet layer* or the *network layer*, provides the "virtual network" image of an internet (this layer shields the higher levels from the physical network architecture below it). Internet Protocol (IP) is the most important protocol in this layer. It is a connectionless protocol that does not assume reliability from lower layers. IP does *not* provide reliability, flow control, or error recovery. These functions must be provided at a higher level.

IP provides a routing function that attempts to deliver transmitted messages to their destination. We discuss IP in detail in Chapter 3, "Internetworking protocols" on page 67. A message unit in an IP network is called an *IP datagram*. This is the basic unit of information transmitted across TCP/IP networks. Other internetwork-layer protocols are IP, ICMP, IGMP, ARP, and RARP.

Network interface layer

The network interface layer, also called the *link layer* or the *data-link layer*, is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer. Examples are IEEE 802.2, X.25 (which is reliable in itself), ATM, FDDI, and even SNA. We discuss some physical networks and interfaces in Chapter 2, "Network interfaces" on page 29.

TCP/IP specifications do not describe or standardize any network-layer protocols per se; they only standardize ways of accessing those protocols from the internetwork layer.

A more detailed layering model is included in Figure 1-3.

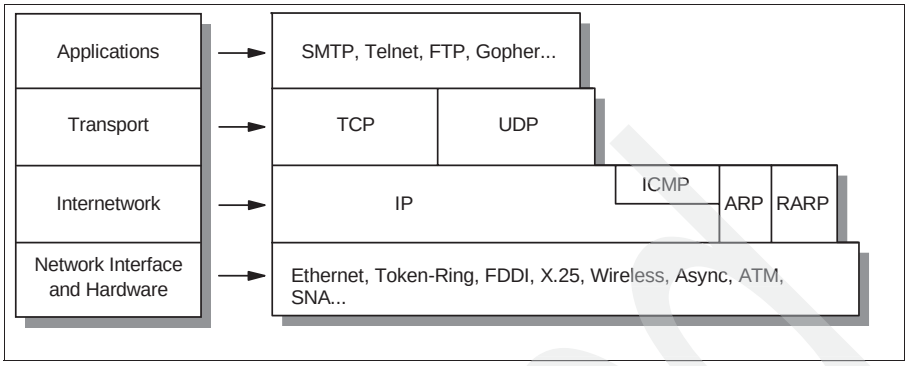


Figure 1-3 Detailed architectural model

1.1.3 TCP/IP applications

The highest-level protocols within the TCP/IP protocol stack are application protocols. They communicate with applications on other internet hosts and are the user-visible interface to the TCP/IP protocol suite.

All application protocols have some characteristics in common:

- They can be user-written applications or applications standardized and shipped with the TCP/IP product. Indeed, the TCP/IP protocol suite includes application protocols such as:
  - Telnet for interactive terminal access to remote internet hosts
  - File Transfer Protocol (FTP) for high-speed disk-to-disk file transfers
  - Simple Mail Transfer Protocol (SMTP) as an internet mailing system

  These are some of the most widely implemented application protocols, but many others exist. Each particular TCP/IP implementation will include a lesser or greater set of application protocols.
- They use either UDP or TCP as a transport mechanism. Remember that UDP is unreliable and offers no flow-control, so in this case, the application has to provide its own error recovery, flow control, and congestion control functionality. It is often easier to build applications on top of TCP because it is a reliable stream, connection-oriented, congestion-friendly, flow control-enabled protocol. As a result, most application protocols will use TCP, but there are applications built on UDP to achieve better performance through increased protocol efficiencies.
- Most applications use the client/server model of interaction.

The client/server model

TCP is a peer-to-peer, connection-oriented protocol. There are no master/subordinate relationships. The applications, however, typically use a client/server model for communications, as demonstrated in Figure 1-4.

A *server* is an application that offers a service to internet users. A *client* is a requester of a service. An application consists of both a server and a client part, which can run on the same or on different systems. Users usually invoke the client part of the application, which builds a *request* for a particular service and sends it to the server part of the application using TCP/IP as a transport vehicle.

The server is a program that receives a request, performs the required service, and sends back the results in a *reply*. A server can usually deal with multiple requests and multiple requesting clients at the same time.

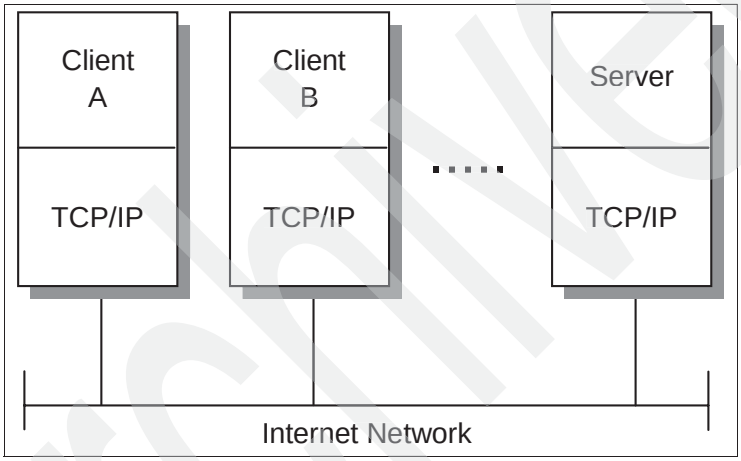


Figure 1-4 The client/server model of applications

Most servers wait for requests at a *well-known port* so that their clients know to which port (and in turn, which application) they must direct their requests. The client typically uses an arbitrary port called an *ephemeral port* for its communication. Clients that want to communicate with a server that does not use a well-known port must have another mechanism for learning to which port they must address their requests. This mechanism might employ a registration service such as portmap, which does use a well-known port.

For detailed information about TCP/IP application protocols, refer to Part 2, “TCP/IP application protocols” on page 405.

Bridges, routers, and gateways

There are many ways to provide access to other networks. In an internetwork, this done with *routers*. In this section, we distinguish between a router, a bridge, and a gateway for allowing remote network access:

Bridge

Interconnects LAN segments at the network interface layer level and forwards frames between them. A bridge performs the function of a MAC relay, and is independent of any higher layer protocol (including the logical link protocol). It provides MAC layer protocol conversion, if required.

A bridge is said to be *transparent* to IP. That is, when an IP host sends an IP datagram to another host on a network connected by a bridge, it sends the datagram directly to the host and the datagram "crosses" the bridge without the sending IP host being aware of it.

Router

Interconnects networks at the internetwork layer level and routes packets between them. The router must understand the addressing structure associated with the networking protocols it supports and take decisions on whether, or how, to forward packets. Routers are able to select the best transmission paths and optimal packet sizes. The basic routing function is implemented in the IP layer of the TCP/IP protocol stack, so any host or workstation running TCP/IP over more than one interface could, in theory and also with most of today's TCP/IP implementations, forward IP datagrams. However, dedicated routers provide much more sophisticated routing than the minimum functions implemented by IP.

Because IP provides this basic routing function, the term "IP router," is often used. Other, older terms for router are "IP gateway," "Internet gateway," and "gateway." The term *gateway* is now normally used for connections at a higher layer than the internetwork layer.

A router is said to be *visible* to IP. That is, when a host sends an IP datagram to another host on a network connected by a router, it sends the datagram to the router so that it can forward it to the target host.

Gateway Interconnects networks at higher layers than bridges and routers. A gateway usually supports address mapping from one network to another, and might also provide transformation of the data between the environments to support end-to-end application connectivity. Gateways typically limit the interconnectivity of two networks to a subset of the application protocols supported on either one. For example, a VM host running TCP/IP can be used as an SMTP/RSCS mail gateway.

Note: The term "gateway," when used in this sense, is *not* synonymous with "IP gateway."

A gateway is said to be *opaque* to IP. That is, a host cannot send an IP datagram through a gateway; it can only send it *to* a gateway. The higher-level protocol information carried by the datagrams is then passed on by the gateway using whatever networking architecture is used on the other side of the gateway.

Closely related to routers and gateways is the concept of a *firewall*, or *firewall gateway*, which is used to restrict access from the Internet or some untrusted network to a network or group of networks controlled by an organization for security reasons. See 22.3, "Firewalls" on page 794 for more information about firewalls.

1.2 The roots of the Internet

Networks have become a fundamental, if not the most important, part of today's information systems. They form the backbone for information sharing in enterprises, governmental groups, and scientific groups. That information can take several forms. It can be notes and documents, data to be processed by another computer, files sent to colleagues, and multimedia data streams.

A number of networks were installed in the late 1960s and 1970s, when network design was the "state of the art" topic of computer research and sophisticated implementers. It resulted in multiple networking models such as packet-switching technology, collision-detection local area networks, hierarchical networks, and many other excellent communications technologies.

The result of all this great know-how was that any group of users could find a physical network and an architectural model suitable for their specific needs. This ranges from inexpensive asynchronous lines with no other error recovery

than a bit-per-bit parity function, through full-function wide area networks (public or private) with reliable protocols such as public packet-switching networks or private SNA networks, to high-speed but limited-distance local area networks.

The down side of the development of such heterogeneous protocol suites is the rather painful situation where one group of users wants to extend its information system to another group of users who have implemented a different network technology and different networking protocols. As a result, even if they could agree on some network technology to physically interconnect the two environments, their applications (such as mailing systems) would still not be able to communicate with each other because of different application protocols and interfaces.

This situation was recognized in the early 1970s by a group of U.S. researchers funded by the *Defense Advanced Research Projects Agency* (DARPA). Their work addressed *internetworking,* or the interconnection of networks. Other official organizations became involved in this area, such as ITU-T (formerly CCITT) and ISO. The main goal was to define a set of protocols, detailed in a well-defined suite, so that applications would be able to communicate with other applications, regardless of the underlying network technology or the operating systems where those applications run.

The official organization of these researchers was the ARPANET Network Working Group, which had its last general meeting in October 1971. DARPA continued its research for an internetworking protocol suite, from the early *Network Control Program* (NCP) host-to-host protocol to the TCP/IP protocol suite, which took its current form around 1978. At that time, DARPA was well known for its pioneering of packet-switching over radio networks and satellite channels. The first real implementations of the *Internet* were found around 1980 when DARPA started converting the machines of its research network (ARPANET) to use the new TCP/IP protocols. In 1983, the transition was completed and DARPA demanded that *all* computers willing to connect to its ARPANET use TCP/IP.

DARPA also contracted Bolt, Beranek, and Newman (BBN) to develop an implementation of the TCP/IP protocols for Berkeley UNIX® on the VAX and funded the University of California at Berkeley to distribute the code free of charge with their UNIX operating system. The first release of the *Berkeley Software Distribution* (BSD) to include the TCP/IP protocol set was made available in 1983 (4.2BSD). From that point on, TCP/IP spread rapidly among universities and research centers and has become the standard communications subsystem for all UNIX connectivity. The second release (4.3BSD) was distributed in 1986, with updates in 1988 (4.3BSD Tahoe) and 1990 (4.3BSD Reno). 4.4BSD was released in 1993. Due to funding constraints, 4.4BSD was

the last release of the BSD by the Computer Systems Research Group of the University of California at Berkeley.

As TCP/IP internetworking spread rapidly, new wide area networks were created in the U.S. and connected to ARPANET. In turn, other networks in the rest of the world, not necessarily based on the TCP/IP protocols, were added to the set of interconnected networks. The result is what is described as *the Internet*. We describe some examples of the different networks that have played key roles in this development in the next sections.

1.2.1 ARPANET

Sometimes referred to as the “grand-daddy of packet networks,” the ARPANET was built by DARPA (which was called ARPA at that time) in the late 1960s to accommodate research equipment on packet-switching technology and to allow resource sharing for the Department of Defense's contractors. The network interconnected research centers, some military bases, and government locations. It soon became popular with researchers for collaboration through electronic mail and other services. It was developed into a research utility run by the Defense Communications Agency (DCA) by the end of 1975 and split in 1983 into MILNET for interconnection of military sites and ARPANET for interconnection of research sites. This formed the beginning of the “capital I” Internet.

In 1974, the ARPANET was based on 56 Kbps leased lines that interconnected *packet-switching nodes* (PSN) scattered across the continental U.S. and western Europe. These were minicomputers running a protocol known as *1822* (after the number of a report describing it) and dedicated to the packet-switching task. Each PSN had at least two connections to other PSNs (to allow alternate routing in case of circuit failure) and up to 22 ports for user computer (*host*) connections. These 1822 systems offered reliable, flow-controlled delivery of a packet to a destination node. This is the reason why the original NCP protocol was a rather simple protocol. It was replaced by the TCP/IP protocols, which do not assume the reliability of the underlying network hardware and can be used on other-than-1822 networks. This 1822 protocol did not become an industry standard, so DARPA decided later to replace the 1822 packet switching technology with the *CCITT X.25* standard.

Data traffic rapidly exceeded the capacity of the 56 Kbps lines that made up the network, which were no longer able to support the necessary throughput. Today the ARPANET has been replaced by new technologies in its role of backbone on the research side of the connected Internet (see NSFNET later in this chapter), while MILNET continues to form the backbone of the military side.

1.2.2 NSFNET

NSFNET, the National Science Foundation (NSF) Network, is a three-level internetwork in the United States consisting of:

- The backbone: A network that connects separately administered and operated mid-level networks and NSF-funded supercomputer centers. The backbone also has transcontinental links to other networks such as EBONE, the European IP backbone network.
- Mid-level networks: Three kinds of networks (regional, discipline-based, and supercomputer consortium networks).
- Campus networks: Whether academic or commercial, connected to the mid-level networks.

Over the years, the NSF upgraded its backbone to meet the increasing demands of its clients:

- First backbone: Originally established by the NSF as a communications network for researchers and scientists to access the NSF supercomputers, the first NSFNET backbone used six DEC LSI/11 microcomputers as packet switches, interconnected by 56 Kbps leased lines. A primary interconnection between the NSFNET backbone and the ARPANET existed at Carnegie Mellon, which allowed routing of datagrams between users connected to each of those networks.
- Second backbone: The need for a new backbone appeared in 1987, when the first one became overloaded within a few months (estimated growth at that time was 100% per year). The NSF and MERIT, Inc., a computer network consortium of eight state-supported universities in Michigan, agreed to develop and manage a new, higher-speed backbone with greater transmission and switching capacities. To manage it, they defined the *Information Services* (IS), which is comprised of an Information Center and a Technical Support Group. The Information Center is responsible for information dissemination, information resource management, and electronic communication. The Technical Support Group provides support directly to the field. The purpose of this is to provide an integrated information system with easy-to-use-and-manage interfaces accessible from any point in the network supported by a full set of training services.

  Merit and NSF conducted this project in partnership with IBM and MCI. IBM provided the software, packet-switching, and network-management equipment, while MCI provided the long-distance transport facilities. Installed in 1988, the new network initially used 448 Kbps leased circuits to interconnect 13 *nodal switching systems* (NSSs), supplied by IBM. Each NSS was composed of nine IBM RISC systems (running an IBM version of 4.3BSD UNIX) loosely coupled by two IBM token-ring networks (for redundancy). One

Integrated Digital Network Exchange (IDNX) supplied by IBM was installed at each of the 13 locations, to provide:

- Dynamic alternate routing
- Dynamic bandwidth allocation

- Third backbone: In 1989, the NSFNET backbone circuits topology was reconfigured after traffic measurements and the speed of the leased lines increased to T1 (1.544 Mbps) using primarily fiber optics.

  Due to the constantly increasing need for improved packet switching and transmission capacities, three NSSs were added to the backbone and the link speed was upgraded. The migration of the NSFNET backbone from T1 to T3 (45 Mbps) was completed in late 1992. The subsequent migration to gigabit levels has already started and is continuing today.

In April 1995, the U.S. government discontinued its funding of NSFNET. This was, in part, a reaction to growing commercial use of the network. About the same time, NSFNET gradually migrated the main backbone traffic in the U.S. to commercial network service providers, and NSFNET reverted to being a network for the research community. The main backbone network is now run in cooperation with MCI and is known as the vBNS (very high speed Backbone Network Service).

NSFNET has played a key role in the development of the Internet. However, many other networks have also played their part and also make up a part of the Internet today.

1.2.3 Commercial use of the Internet

In recent years the Internet has grown in size and range at a greater rate than anyone could have predicted. A number of key factors have influenced this growth. Some of the most significant milestones have been the free distribution of Gopher in 1991, the first posting, also in 1991, of the specification for hypertext and, in 1993, the release of Mosaic, the first graphics-based browser. Today the vast majority of the hosts now connected to the Internet are of a commercial nature. This is an area of potential and actual conflict with the initial aims of the Internet, which were to foster open communications between academic and research institutions. However, the continued growth in commercial use of the Internet is inevitable, so it will be helpful to explain how this evolution is taking place.

One important initiative to consider is that of the *Acceptable Use Policy* (AUP). The first of these policies was introduced in 1992 and applies to the use of NSFNET. At the heart of this AUP is a commitment “to support open research and education.” Under “Unacceptable Uses” is a prohibition of “use for for-profit

activities," unless covered by the General Principle or as a specifically acceptable use. However, in spite of this apparently restrictive stance, the NSFNET was increasingly used for a broad range of activities, including many of a commercial nature, before reverting to its original objectives in 1995.

The provision of an AUP is now commonplace among Internet service providers, although the AUP has generally evolved to be more suitable for commercial use. Some networks still provide services free of any AUP.

Let us now focus on the Internet service providers who have been most active in introducing commercial uses to the Internet. Two worth mentioning are PSINet and UUNET, which began in the late 1980s to offer Internet access to both businesses and individuals. The California-based CERFnet provided services free of any AUP. An organization to interconnect PSINet, UUNET, and CERFnet was formed soon after, called the Commercial Internet Exchange (CIX), based on the understanding that the traffic of any member of one network may flow without restriction over the networks of the other members. As of July 1997, CIX had grown to more than 146 members from all over the world, connecting member internets. At about the same time that CIX was formed, a non-profit company, Advance Network and Services (ANS), was formed by IBM, MCI, and Merit, Inc. to operate T1 (subsequently T3) backbone connections for NSFNET. This group was active in increasing the commercial presence on the Internet.

ANS formed a commercially oriented subsidiary called ANS CO+RE to provide linkage between commercial customers and the research and education domains. ANS CO+RE provides access to NSFNET as well as being linked to CIX. In 1995 ANS was acquired by America Online.

In 1995, as the NSFNET was reverting to its previous academic role, the architecture of the Internet changed from having a single dominant backbone in the U.S. to having a number of commercially operated backbones. In order for the different backbones to be able to exchange data, the NSF set up four Network Access Points (NAPs) to serve as data interchange points between the backbone service providers.

Another type of interchange is the Metropolitan Area Ethernet (MAE). Several MAEs have been set up by Metropolitan Fiber Systems (MFS), who also have their own backbone network. NAPs and MAEs are also referred to as public exchange points (IXPs). Internet service providers (ISPs) typically will have connections to a number of IXPs for performance and backup. For a current listing of IXPs, consult the Exchange Point at:

`http://www.ep.net`

Similar to CIX in the United States, European Internet providers formed the RIPE (Réseaux IP Européens) organization to ensure technical and administrative

coordination. RIPE was formed in 1989 to provide a uniform IP service to users throughout Europe. Today, the largest Internet backbones run at OC48 (2.4 Gbps) or OC192 (9.6 Gbps).

1.2.4 Internet2

The success of the Internet and the subsequent frequent congestion of the NSFNET and its commercial replacement led to some frustration among the research community who had previously enjoyed exclusive use of the Internet. The university community, therefore, together with government and industry partners, and encouraged by the funding component of the Next Generation Internet (NGI) initiative, have formed the *Internet2* project.

The NGI initiative is a federal research program that is developing advanced networking technologies, introducing revolutionary applications that require advanced networking technologies and demonstrating these technological capabilities on high-speed testbeds.

Mission

The Internet2 mission is to facilitate and coordinate the development, operation, and technology transfer of advanced, network-based applications and network services to further U.S. leadership in research and higher education and accelerate the availability of new services and applications on the Internet.

Internet2 has the following goals:

- Demonstrate new applications that can dramatically enhance researchers' ability to collaborate and conduct experiments.
- Demonstrate enhanced delivery of education and other services (for instance, health care, environmental monitoring, and so on) by taking advantage of *virtual proximity* created by an advanced communications infrastructure.
- Support development and adoption of advanced applications by providing middleware and development tools.
- Facilitate development, deployment, and operation of an affordable communications infrastructure, capable of supporting differentiated quality of service (QoS) based on application requirements of the research and education community.
- Promote experimentation with the next generation of communications technologies.
- Coordinate adoption of agreed working standards and common practices among participating institutions to ensure end-to-end quality of service and interoperability.

- Catalyze partnerships with governmental and private sector organizations.
- Encourage transfer of technology from Internet2 to the rest of the Internet.
- Study the impact of new infrastructure, services, and applications on higher education and the Internet community in general.

Internet2 participants

Internet2 has 180 participating universities across the United States. Affiliate organizations provide the project with valuable input. All participants in the Internet2 project are members of the University Corporation for Advanced Internet Development (UCAID).

In most respects, the partnership and funding arrangements for Internet2 will parallel those of previous joint networking efforts of academia and government, of which the NSFnet project is a very successful example. The United States government will participate in Internet2 through the NGI initiative and related programs.

Internet2 also joins with corporate leaders to create the advanced network services necessary to meet the requirements of broadband, networked applications. Industry partners work primarily with campus-based and regional university teams to provide the services and products needed to implement the applications developed by the project. Major corporations currently participating in Internet2 include Alcatel, Cisco Systems, IBM, Nortel Networks, Sprint, and Sun Microsystems™. Additional support for Internet2 comes from collaboration with non-profit organizations working in research and educational networking. Affiliate organizations committed to the project include MCNC, Merit, National Institutes of Health (NIH), and the State University System of Florida.

For more information about Internet2, see their Web page at:

`http://www.internet2.edu`

Archived

1.2.5 The Open Systems Interconnection (OSI) Reference Model

The OSI (Open Systems Interconnect) Reference Model (ISO 7498) defines a seven-layer model of data communication with physical transport at the lower layer and application protocols at the upper layers. This model, shown in Figure 1-5, is widely accepted as a basis for the understanding of how a network protocol stack should operate and as a reference tool for comparing network stack implementation.

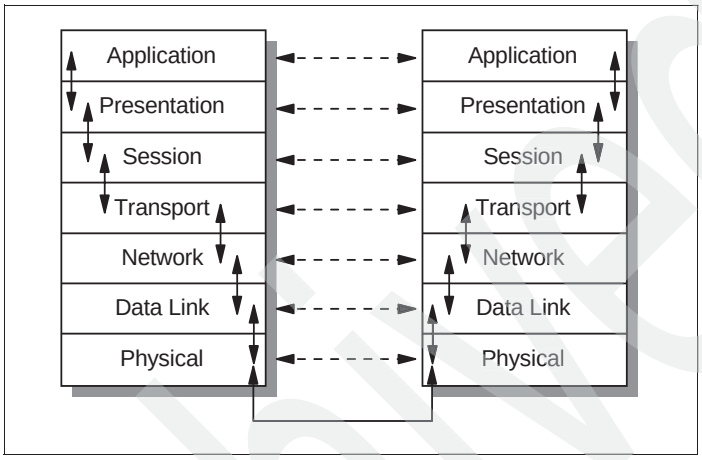


Figure 1-5 The OSI Reference Model

Each layer provides a set of functions to the layer above and, in turn, relies on the functions provided by the layer below. Although messages can only pass vertically through the stack from layer to layer, from a logical point of view, each layer communicates directly with its peer layer on other nodes.

The seven layers are:

Application	Network applications such as terminal emulation and file transfer
Presentation	Formatting of data and encryption
Session	Establishment and maintenance of sessions
Transport	Provision of reliable and unreliable end-to-end delivery
Network	Packet delivery, including routing
Data Link	Framing of units of information and error checking
Physical	Transmission of bits on the physical hardware

In contrast to TCP/IP, the OSI approach started from a clean slate and defined standards, adhering tightly to their own model, using a formal committee process

without requiring implementations. Internet protocols use a less formal engineering approach, where anybody can propose and comment on Request for Comments, known as RFC, and implementations are required to verify feasibility. The OSI protocols developed slowly, and because running the full protocol stack is resource intensive, they have not been widely deployed, especially in the desktop and small computer market. In the meantime, TCP/IP and the Internet were developing rapidly, with deployment occurring at a very high rate.

1.3 TCP/IP standards

TCP/IP has been popular with developers and users alike because of its inherent openness and perpetual renewal. The same holds true for the Internet as an open communications network. However, this openness could easily turn into something that can help you and hurt you if it were not controlled in some way. Although there is no overall governing body to issue directives and regulations for the Internet—control is mostly based on mutual cooperation—the Internet Society (ISOC) serves as the standardizing body for the Internet community. It is organized and managed by the Internet Architecture Board (IAB).

The IAB itself relies on the Internet Engineering Task Force (IETF) for issuing new standards, and on the Internet Assigned Numbers Authority (IANA) for coordinating values shared among multiple protocols. The RFC Editor is responsible for reviewing and publishing new standards documents.

The IETF itself is governed by the Internet Engineering Steering Group (IESG) and is further organized in the form of Areas and Working Groups where new specifications are discussed and new standards are propsoed.

The Internet Standards Process, described in RFC 2026, The Internet Standards Process, Revision 3, is concerned with all protocols, procedures, and conventions that are used in or by the Internet, whether or not they are part of the TCP/IP protocol suite.

The overall goals of the Internet Standards Process are:

- Technical excellence
- Prior implementation and testing
- Clear, concise, and easily understood documentation
- Openness and fairness
- Timeliness

The process of standardization is summarized as follows:

- In order to have a new specification approved as a standard, applicants have to submit that specification to the IESG where it will be discussed and reviewed for technical merit and feasibility and also published as an Internet draft document. This should take no shorter than two weeks and no longer than six months.
- After the IESG reaches a positive conclusion, it issues a last-call notification to allow the specification to be reviewed by the whole Internet community.
- After the final approval by the IESG, an Internet draft is recommended to the Internet Engineering Taskforce (IETF), another subsidiary of the IAB, for inclusion into the Standards Track and for publication as a Request for Comments (see 1.3.1, “Request for Comments (RFC)” on page 22).
- Once published as an RFC, a contribution may advance in status as described in 1.3.2, “Internet standards” on page 24. It may also be revised over time or phased out when better solutions are found.
- If the IESG does not approve of a new specification after, or if a document has remained unchanged within, six months of submission, it will be removed from the Internet drafts directory.

1.3.1 Request for Comments (RFC)

The Internet protocol suite is still evolving through the mechanism of *Request for Comments* (RFC). New protocols (mostly application protocols) are being designed and implemented by researchers, and are brought to the attention of the Internet community in the form of an Internet draft (ID).[1] The largest source of IDs is the Internet Engineering Task Force (IETF), which is a subsidiary of the IAB. However, anyone can submit a memo proposed as an ID to the RFC Editor. There are a set of rules which RFC/ID authors must follow in order for an RFC to be accepted. These rules are themselves described in an RFC (RFC 2223), which also indicates how to submit a proposal for an RFC.

After an RFC has been published, all revisions and replacements are published as new RFCs. A new RFC that revises or replaces an existing RFC is said to “update” or to “obsolete” that RFC. The existing RFC is said to be “updated by” or “obsoleted by” the new one. For example RFC 1542, which describes the BOOTP protocol, is a “second edition,” being a revision of RFC 1532 and an amendment to RFC 951. RFC 1542 is therefore labelled like this: “Obsoletes RFC 1532; Updates RFC 951." Consequently, there is never any confusion over

[1] Some of these protocols can be described as impractical at best. For instance, RFC 1149 (dated 1990 April 1) describes the transmission of IP datagrams by carrier pigeon and RFC 1437 (dated 1993 April 1) describes the transmission of people by electronic mail.

whether two people are referring to different versions of an RFC, because there is never more than one current version.

Some RFCs are described as *information documents*, while others describe Internet protocols. The Internet Architecture Board (IAB) maintains a list of the RFCs that describe the protocol suite. Each of these is assigned a *state* and a *status*.

An Internet protocol can have one of the following states:

Standard The IAB has established this as an official protocol for the Internet. These are separated into two groups:

- IP protocol and above, protocols that apply to the whole Internet
- Network-specific protocols, generally specifications of how to do IP on particular types of networks

Draft standard The IAB is actively considering this protocol as a possible standard protocol. Substantial and widespread testing and comments are desired. Submit comments and test results to the IAB. There is a possibility that changes will be made in a draft protocol before it becomes a standard.

Proposed standard These are protocol proposals that might be considered by the IAB for standardization in the future. Implementations and testing by several groups are desirable. Revision of the protocol is likely.

Experimental A system should not implement an experimental protocol unless it is participating in the experiment and has coordinated its use of the protocol with the developer of the protocol.

Informational Protocols developed by other standard organizations, or vendors, or that are for other reasons outside the purview of the IAB may be published as RFCs for the convenience of the Internet community as informational protocols. Such protocols might, in some cases, also be recommended for use on the Internet by the IAB.

Historic These are protocols that are unlikely to ever become standards in the Internet either because they have been superseded by later developments or due to lack of interest.

Archived

Protocol status can be any of the following:

Required	A system must implement the required protocols.
Recommended	A system should implement the recommended protocol.
Elective	A system may or may not implement an elective protocol. The general notion is that if you are going to do something like this, you must do exactly this.
Limited use	These protocols are for use in limited circumstances. This may be because of their experimental state, specialized nature, limited functionality, or historic state.
Not recommended	These protocols are not recommended for general use. This may be because of their limited functionality, specialized nature, or experimental or historic state.

1.3.2 Internet standards

Proposed standard, draft standard, and standard protocols are described as being on the *Internet Standards Track*. When a protocol reaches the standard state, it is assigned a standard (STD) number. The purpose of STD numbers is to clearly indicate which RFCs describe Internet standards. STD numbers reference multiple RFCs when the specification of a standard is spread across multiple documents. Unlike RFCs, where the number refers to a specific document, STD numbers do not change when a standard is updated. STD numbers do not, however, have version numbers because all updates are made through RFCs and the RFC numbers are unique. Therefore, to clearly specify which version of a standard one is referring to, the standard number and all of the RFCs that it includes should be stated. For instance, the Domain Name System (DNS) is STD 13 and is described in RFCs 1034 and 1035. To reference the standard, a form such as “STD-13/RFC1034/RFC1035” should be used.

For some Standards Track RFCs, the status category does not always contain enough information to be useful. It is therefore supplemented, notably for routing protocols, by an *applicability statement*, which is given either in STD 1 or in a separate RFC.

References to the RFCs and to STD numbers will be made throughout this book, because they form the basis of all TCP/IP protocol implementations.

The following Internet standards are of particular importance:

- STD 1 – Internet Official Protocol Standards

  This standard gives the state and status of each Internet protocol or standard and defines the meanings attributed to each state or status. It is issued by the IAB approximately quarterly. At the time of writing, this standard is in RFC 3700.

- STD 2 – Assigned Internet Numbers

  This standard lists currently assigned numbers and other protocol parameters in the Internet protocol suite. It is issued by the Internet Assigned Numbers Authority (IANA). The current edition at the time of writing is RFC 3232.

- STD 3 – Host Requirements

  This standard defines the requirements for Internet host software (often by reference to the relevant RFCs). The standard comes in three parts:

  – RFC 1122 – Requirements for Internet hosts – communications layer
  – RFC 1123 – Requirements for Internet hosts – application and support
  – RFC 2181 – Clarifications to the DNS Specification

- STD 4 – Router Requirements

  This standard defines the requirements for IPv4 Internet gateway (router) software. It is defined in RFC 1812 – Requirements for IPv4 Routers.

For Your Information (FYI)

A number of RFCs that are intended to be of wide interest to Internet users are classified as *For Your Information* (FYI) documents. They frequently contain introductory or other helpful information. Like STD numbers, an FYI number is not changed when a revised RFC is issued. Unlike STDs, FYIs correspond to a single RFC document. For example, FYI 4 - FYI on Questions and Answers - Answers to Commonly asked “New Internet User” Questions, is currently in its fifth edition. The RFC numbers are 1177, 1206, 1325 and 1594, and 2664.

Obtaining RFCs

RFC and ID documents are available publicly and online and best obtained from the IETF Web site:

`http://www.ietf.org`

A complete list of current Internet Standards can be found in RFC 3700 – Internet Official Protocol Standards.

1.4 Future of the Internet

Trying to predict the future of the Internet is not an easy task. Few would have imagined, even five years ago, the extent to which the Internet has now become a part of everyday life in business, homes, and schools. There are a number of things, however, about which we can be fairly certain.

1.4.1 Multimedia applications

Bandwidth requirements will continue to increase at massive rates; not only is the number of Internet users growing rapidly, but the applications being used are becoming more advanced and therefore consume more bandwidth. New technologies such as dense wave division multiplexing (DWDM) are emerging to meet these high bandwidth demands being placed on the Internet.

Much of this increasing demand is attributable to the increased use of multimedia applications. One example is that of Voice over IP technology. As this technology matures, we are almost certain to see a sharing of bandwidth between voice and data across the Internet. This raises some interesting questions for phone companies. The cost to a user of an Internet connection between Raleigh, NC and Santiago, Chile is the same as a connection within Raleigh, not so for a traditional phone connection. Inevitably, voice conversations will become video conversations as phone calls become video conferences.

Today, it is possible to hear radio stations from almost any part of the globe through the Internet with FM quality. We can watch television channels from all around the world, leading to the clear potential of using the Internet as the vehicle for delivering movies and all sorts of video signals to consumers everywhere. It all comes at a price, however, as the infrastructure of the Internet must adapt to such high bandwidth demands.

1.4.2 Commercial use

The Internet has been through an explosion in terms of commercial use. Today, almost all large business depend on the Internet, whether for marketing, sales, customer service, or employee access. These trends are expected to continue. Electronic stores will continue to flourish by providing convenience to customers that do not have time to make their way to traditional stores.

Businesses will rely more and more on the Internet as a means for communicating branches across the globe. With the popularity of virtual private networks (VPNs), businesses can securely conduct their internal business over a wide area using the Internet; employees can work from home offices yielding a

virtual office environment. Virtual meetings probably will be common occurrences.

1.4.3 The wireless Internet

Perhaps the most widespread growth in the use of the Internet, however, is that of wireless applications. Recently, there has been an incredible focus on the enablement of wireless and pervasive computing. This focus has been largely motivated by the convenience of wireless connectivity. For example, it is impractical to physically connect a mobile workstation, which by definition, is free to roam. Constraining such a workstation to some physical geography simply defeats the purpose. In other cases, wired connectivity simply is not feasible. Examples include the ruins of Macchu Picchu or offices in the Sistine Chapel. In these circumstances, fixed workstations also benefit from otherwise unavailable network access.

Protocols such as Bluetooth, IEEE 802.11, and Wireless Application Protocol (WAP) are paving the way toward a wireless Internet. While the personal benefits of such access are quite advantageous, even more appealing are the business applications that are facilitated by such technology. Every business, from factories to hospitals, could enhance their respective services. Wireless devices will become standard equipment in vehicles, not only for the personal enjoyment of the driver, but also for the flow of maintenance information to your favorite automobile mechanic. The applications are limitless.

1.5 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 2026 – The Internet Standards Process -- Revision 3 (October 1996)
- RFC 2223 – Instructions to RFC (October 1997)
- RFC 2900 – Internet Official Protocol Standards (August 2001)
- RFC 3232 – Assigned Numbers: RFC 1700 is Replaced by an On-line Database (January 2002)

Archived

2

Network interfaces

This chapter provides an overview of the protocols and interfaces that allow TCP/IP traffic to flow over various kinds of physical networks. TCP/IP, as an internetwork protocol suite, can operate over a vast number of physical networks. The most common and widely used of these protocols is, of course, Ethernet. The number of network protocols that have provisions for natively supporting IP is clearly beyond the scope of this redbook. However, we provide a summary of some of the different networks most commonly used with TCP/IP. Although this chapter primarily discusses the data link layer of the OSI model (see 1.2.5, “The Open Systems Interconnection (OSI) Reference Model” on page 20), it also provides some relevant physical layer technology.

© Copyright IBM Corp. 1989-2006. All rights reserved.

2.1 Ethernet and IEEE 802 local area networks (LANs)

Two frame formats (or standards) can be used on the Ethernet coaxial cable:

- The standard issued in 1978 by Xerox Corporation, Intel® Corporation, and Digital Equipment Corporation, usually called *Ethernet* (or *DIX* Ethernet)
- The international IEEE 802.3 standard, a more recently defined standard

See Figure 2-1 for more details.

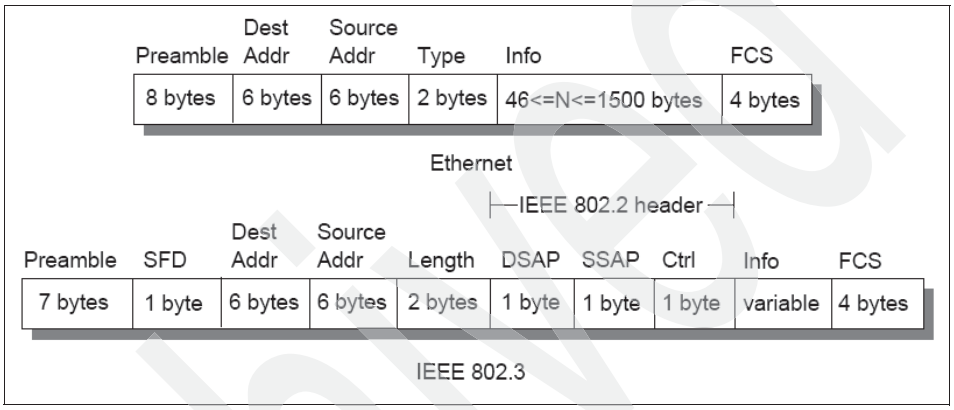


Figure 2-1 ARP: Frame formats for Ethernet and IEEE 802.3

The difference between the two standards is in the use of one of the header fields, which contains a protocol-type number for Ethernet and the length of the data in the frame for IEEE 802.3:

- The type field in Ethernet is used to distinguish between different protocols running on the coaxial cable, and allows their coexistence on the same physical cable.
- The maximum length of an Ethernet frame is 1526 bytes. This means a data field length of up to 1500 bytes. The length of the 802.3 data field is also limited to 1500 bytes for 10 Mbps networks, but is different for other transmission speeds.
- In the 802.3 MAC frame, the length of the data field is indicated in the 802.3 header. The type of protocol it carries is then indicated in the 802.2 header (higher protocol level; see Figure 2-1). In practice, however, both frame formats can coexist on the same physical coaxial. This is done by using protocol type numbers (type field) greater than 1500 in the Ethernet frame. However, different device drivers are needed to handle each of these formats.

Therefore, for all practical purposes, the Ethernet physical layer and the IEEE 802.3 physical layer are compatible. However, the Ethernet data link layer and the IEEE 802.3/802.2 data link layer are incompatible.

The 802.2 Logical Link Control (LLC) layer above IEEE 802.3 uses a concept known as *link service access point* (LSAP), which uses a 3-byte header, where DSAP and SSAP stand for destination and source service access point, respectively. Numbers for these fields are assigned by an IEEE committee (see Figure 2-2).

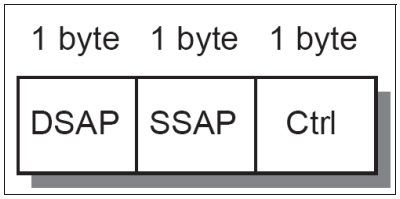


Figure 2-2 ARP: IEEE 802.2 LSAP header

Due to a growing number of applications using IEEE 802 as lower protocol layers, an extension was made to the IEEE 802.2 protocol in the form of the Subnetwork Access Protocol (SNAP) (see Figure 2-3). It is an extension to the LSAP header in Figure 2-2, and its use is indicated by the value 170 in both the SSAP and DSAP fields of the LSAP frame Figure 2-3.

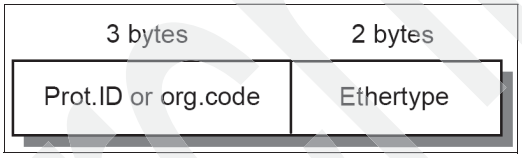


Figure 2-3 ARP: IEEE 802.2 SNAP header

In the evolution of TCP/IP, standards were established that describe the encapsulation of IP and ARP frames on these networks:

- Introduced in 1984, RFC 894 – Standard for the Transmission of IP Datagrams over Ethernet Networks specifies only the use of Ethernet type of networks. The values assigned to the type field are:
  - 2048 (hex 0800), for IP datagrams
  - 2054 (hex 0806), for ARP datagrams

Archived

- Introduced in 1985, RFC 948 – Two Methods for the Transmission of IP Datagrams over IEEE 802.3 Networks specifies two possibilities:
  - The Ethernet compatible method: The frames are sent on a real IEEE 802.3 network in the same fashion as on an Ethernet network, that is, using the IEEE 802.3 data-length field as the Ethernet type field, thereby violating the IEEE 802.3 rules, but compatible with an Ethernet network.
  - IEEE 802.2/802.3 LLC type 1 format: Using 802.2 LSAP header with IP using the value 6 for the SSAP and DSAP fields.

  The RFC indicates clearly that the IEEE 802.2/802.3 method is the preferred method, that is, that all future IP implementations on IEEE 802.3 networks are supposed to use the second method.
- Introduced in 1987, RFC 1010 – Assigned Numbers (now obsoleted by RFC 3232, dated 2002) notes that as a result of IEEE 802.2 evolution and the need for more Internet protocol numbers, a new approach was developed based on practical experiences exchanged during the August 1986 TCP Vendors Workshop. It states, in an almost completely overlooked part of this RFC, that all IEEE 802.3, 802.4, and 802.5 implementations should use the Subnetwork Access Protocol (SNAP) form of the IEEE 802.2 LLC, with the DSAP and SSAP fields set to 170 (indicating the use of SNAP), with SNAP assigned as follows:
  - 0 (zero) as organization code
  - EtherType field:
    - 2048 (hex 0800), for IP datagrams
    - 2054 (hex 0806), for ARP datagrams
    - 32821 (hex 8035), for RARP datagrams

  These are the same values used in the Ethernet type field.
- In1988, RFC 1042 – Standard for the Transmission of IP Datagrams over IEEE 802 Networks was introduced. Because this new approach (very important for implementations) passed almost unnoticed in a little note of an unrelated RFC, it became quite confusing. As such, in February of 1988 it was repeated in an RFC on its own, RFC 1042, which now obsoletes RFC 948.
- In 1998, RFC 2464 – Transmission of IPv6 Packets over Ethernet Networks was introduced. This extended Ethernet's frame format to allow IPv6 packets to traverse Ethernet networks.

The relevant IBM TCP/IP products implement RFC 894 for DIX Ethernet and RFC 3232 for IEEE 802.3 networks. However, in practical situations, there are still TCP/IP implementations that use the older LSAP method (RFC 948 or 1042).

Such implementations will not communicate with the more recent implementations (such as IBM's).

Also note that the last method covers not only the IEEE 802.3 networks, but also the IEEE 802.4 and 802.5 networks, such as the IBM token-ring LAN.

2.1.1 Gigabit Ethernet

As advances in hardware continue to provide faster transmissions across networks, Ethernet implementations have improved in order to capitalize on the faster speeds. *Fast Ethernet* increased the speed of traditional Ethernet from 10 megabits per second (Mbps) to 100 Mbps. This was further augmented to 1000 Mbps in June of 1998, when the IEEE defined the standard for *Gigabit Ethernet* (IEEE 802.3z). Finally, in 2005, IEEE created the 802.3-2005 standard introduced *10 Gigabit Ethernet*, also referred to as *10GbE*. 10GbE provides transmission speeds of 10 gigabits per second (Gbps), or 10000 Mbps, 10 times the speed of Gigabit Ethernet. However, due to the novelty of 10GbE, there are still limitations on the adapters over which 10GbE can be used, and no one implementation standard has yet gained commercial acceptance.

2.2 Fiber Distributed Data Interface (FDDI)

The FDDI specifications define a family of standards for 100 Mbps fiber optic LANs that provides the physical layer and media access control sublayer of the data link layer, as defined by the ISO/OSI Model. Proposed initially by draft-standard RFC 1188, IP and ARP over FDDI networks became a standard in RFC 1390 (also STD 0036). It defines the encapsulating of IP datagrams and ARP requests and replies in FDDI frames. RFC 2467 extended this standard in order to allow the transmission of IPv6 packets over FDDI networks. Operation on dual MAC stations is described in informational RFC 1329. Figure 2-4 on page 34 shows the related protocol layers.

RFC 1390 states that all frames are transmitted in standard IEEE 802.2 LLC Type 1 Unnumbered Information format, with the DSAP and SSAP fields of the 802.2 header set to the assigned global SAP® value for SNAP (decimal 170). The 24-bit Organization Code in the SNAP header is set to zero, and the remaining 16 bits are the EtherType from Assigned Numbers (see RFC 3232), that is:

- 2048 for IP
- 2054 for ARP

The mapping of 32-bit Internet addresses to 48-bit FDDI addresses is done through the ARP dynamic discovery procedure. The broadcast Internet addresses (whose host address is set to all ones) are mapped to the broadcast FDDI address (all ones).

IP datagrams are transmitted as series of 8-bit bytes using the usual TCP/IP transmission order called *big-endian* or network byte order.

The FDDI MAC specification (ISO 9314-2 - ISO, Fiber Distributed Data Interface - Media Access Control) defines a maximum frame size of 4500 bytes for all frame fields. After taking the LLC/SNAP header into account, and to allow future extensions to the MAC header and frame status fields, the MTU of FDDI networks is set to 4352 bytes.

Refer to the IBM Redbook *Local Area Network Concepts and Products: LAN Architect*, SG24-4753, the first volume of the four-volume series *LAN Concepts and Products*, for more details about the FDDI architecture.

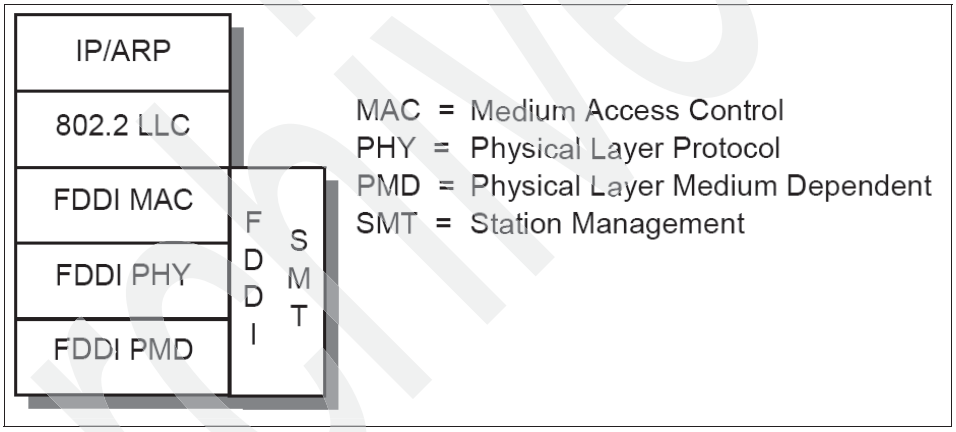


Figure 2-4 IP and ARP over FDDI

2.3 Serial Line IP (SLIP)

The TCP/IP protocol family runs over a variety of network media: IEEE 802.3 and 802.5 LANs, X.25 lines, satellite links, and serial lines. Standards for the encapsulation of IP packets have been defined for many of these networks, but there is no standard for serial lines. SLIP is currently a *de facto* standard, commonly used for point-to-point serial connections running TCP/IP. Even though SLIP is not an Internet standard, it is documented by RFC 1055.

SLIP is just a very simple protocol designed quite a long time ago and is merely a packet framing protocol. It defines a sequence of characters that frame IP packets on a serial line, and nothing more. It does not provide any of the following:

- Addressing: Both computers on a SLIP link need to know each other's IP address for routing purposes. SLIP defines only the encapsulation protocol, not any form of handshaking or link control. Links are manually connected and configured, including the specification of the IP address.
- Packet type identification: SLIP cannot support multiple protocols across a single link; thus, only one protocol can be run over a SLIP connection.
- Error detection/correction: SLIP does no form of frame error detection. The higher-level protocols should detect corrupted packets caused by errors on noisy lines. (IP header and UDP/TCP checksums should be sufficient.) Because it takes so long to retransmit a packet that was altered, it would be efficient if SLIP could provide some sort of simple error correction mechanism of its own.
- Compression: SLIP provides no mechanism for compressing frequently used IP header fields. Many applications over slow serial links tend to be single-user interactive TCP traffic, such as Telnet. This frequently involves small packet sizes and inefficiencies in TCP and IP headers that do not change much between datagrams, but that can have a noticeably detrimental effect on interactive response times.

  However, many SLIP implementations now use *Van Jacobson Header Compression*. This is used to reduce the size of the combined IP and TCP headers from 40 bytes to 3-4 bytes by recording the states of a set of TCP connections at each end of the link and replacing the full headers with encoded updates for the normal case, where many of the fields are unchanged or are incremented by small amounts between successive IP datagrams for a session. RFC 1144 describes this compression.

The SLIP protocol has been essentially replaced by the Point-to-Point Protocol (PPP), as described in the following section.

2.4 Point-to-Point Protocol (PPP)

Point-to-Point Protocol (PPP) is a network-specific standard protocol with STD number 51. Its status is elective, and it is described in RFC 1661 and RFC 1662. The standards defined in these RFCs were later extended to allow IPv6 over PPP, defined in RFC 2472.

There are a large number of *proposed standard protocols*, which specify the operation of PPP over different kinds of point-to-point links. Each has a status of elective. We advise you to consult STD 1 – Internet Official Protocol Standards for a list of PPP-related RFCs that are on the Standards Track.

Point-to-point circuits in the form of asynchronous and synchronous lines have long been the mainstay for data communications. In the TCP/IP world, the de facto standard SLIP protocol (see 2.3, “Serial Line IP (SLIP)” on page 34) has served admirably in this area, and is still in widespread use for dial-up TCP/IP connections. However, SLIP has a number of drawbacks that are addressed by the Point-to-Point Protocol.

PPP has three main components:

- A method for encapsulating datagrams over serial links.
- A *Link Control Protocol (LCP)* for establishing, configuring, and testing the data-link connection.
- A family of *Network Control Protocols* (NCPs) for establishing and configuring different network-layer protocols. PPP is designed to allow the simultaneous use of multiple network-layer protocols.

Before a link is considered to be ready for use by network-layer protocols, a specific sequence of events must happen. The LCP provides a method of establishing, configuring, maintaining, and terminating the connection. LCP goes through the following phases:

1. Link establishment and configuration negotiation: In this phase, link control packets are exchanged and link configuration options are negotiated. After options are agreed on, the link is *open*, but is not necessarily *ready* for network-layer protocols to be started.
2. Link quality determination: This phase is optional. PPP does not specify the policy for determining quality, but does provide low-level tools, such as echo request and reply.
3. Authentication: This phase is optional. Each end of the link authenticates itself with the remote end using authentication methods agreed to during phase 1.
4. Network-layer protocol configuration negotiation: After LCP has finished the previous phase, network-layer protocols can be separately configured by the appropriate NCP.
5. Link termination: LCP can terminate the link at any time. This is usually done at the request of a human user, but can happen because of a physical event.

Archived

2.4.1 Point-to-point encapsulation

A summary of the PPP encapsulation is shown in Figure 2-5.



Figure 2-5 PPP encapsulation frame

The encapsulation fields are defined as follows:

Protocol field The protocol field is one or two octets, and its value identifies the datagram encapsulated in the Information field of the packet. Up-to-date values of the Protocol field are specified in RFC 3232.

Information field The Information field is zero or more octets. The Information field contains the datagram for the protocol specified in the Protocol field. The maximum length for the information field, including padding, but not including the Protocol field, is termed the Maximum Receive Unit (MRU), which defaults to 1500 octets. By negotiation, other values can be used for the MRU.

Padding On transmission, the information field can be padded with an arbitrary number of octets up to the MRU. It is the responsibility of each protocol to distinguish padding octets from real information.

The *IP Control Protocol (IPCP)* is the NCP for IP and is responsible for configuring, enabling, and disabling the IP protocol on both ends of the point-to-point link. The IPCP options negotiation sequence is the same as for LCP, thus allowing the possibility of reusing the code.

One important option used with IPCP is Van Jacobson Header Compression, which is used to reduce the size of the combined IP and TCP headers from 40 bytes to approximately 3-4 by recording the states of a set of TCP connections at each end of the link and replacing the full headers with encoded updates for the normal case, where many of the fields are unchanged or are incremented by small amounts between successive IP datagrams for a session. This compression is described in RFC 1144.

Archived

2.5 Integrated Services Digital Network (ISDN)

This section describes how to use the PPP encapsulation over ISDN point-to-point links. PPP over ISDN is documented by elective RFC 1618. Because the ISDN B-channel is, by definition, a point-to-point circuit, PPP is well suited for use over these links.

The ISDN Basic Rate Interface (BRI) usually supports two B-channels with a capacity of 64 kbps each, and a 16 kbps D-channel for control information. B-channels can be used for voice or data or just for data in a combined way.

The ISDN Primary Rate Interface (PRI) can support many concurrent B-channel links (usually 30) and one 64 Kbps D-channel. The PPP LCP and NCP mechanisms are particularly useful in this situation in reducing or eliminating manual configuration and facilitating ease of communication between diverse implementations. The ISDN D-channel can also be used for sending PPP packets when suitably framed, but is limited in bandwidth and often restricts communication links to a local switch.

PPP treats ISDN channels as bit- or octet-oriented synchronous links. These links must be full-duplex, but can be either dedicated or circuit-switched. PPP presents an octet interface to the physical layer. There is no provision for sub-octets to be supplied or accepted. PPP does not impose any restrictions regarding transmission rate other than that of the particular ISDN channel interface. PPP does not require the use of control signals. When available, using such signals can allow greater functionality and performance.

The definition of various encodings and scrambling is the responsibility of the DTE/DCE equipment in use. While PPP will operate without regard to the underlying representation of the bit stream, lack of standards for transmission will hinder interoperability as surely as lack of data link standards. The D-channel interface requires Non-Return-To-Zero (NRZ) encoding. Therefore, it is recommended that NRZ be used over the B-channel interface. This will allow frames to be easily exchanged between the B- and D-channels. However, when the configuration of the encoding is allowed, NRZ Inverted (NRZI) is recommended as an alternative in order to ensure a minimum ones density where required over the clear B-channel. Implementations that want to interoperate with multiple encodings can choose to detect those encodings automatically. Automatic encoding detection is particularly important for primary rate interfaces to avoid extensive preconfiguration.

Terminal adapters conforming to V.120[1] can be used as a simple interface to workstations. The terminal adapter provides asynchronous-to-synchronous conversion. Multiple B-channels can be used in parallel. V.120 is not interoperable with bit-synchronous links, because V.120 does not provide octet

stuffing to bit stuffing conversion. Despite the fact that HDLC, LAPB, LAPD, and LAPF are nominally distinguishable, multiple methods of framing should not be used concurrently on the same ISDN channel. There is no requirement that PPP recognize alternative framing techniques, or switch between framing techniques without specific configuration. Experience has shown that the LLC Information Element is not reliably transmitted end to end. Therefore, transmission of the LLC-IE should not be relied upon for framing or encoding determination. No LLC-IE values that pertain to PPP have been assigned. Any other values that are received are not valid for PPP links, and can be ignored for PPP service. The LCP recommended sync configuration options apply to ISDN links. The standard LCP sync configuration defaults apply to ISDN links. The typical network connected to the link is likely to have an MRU size of either 1500 or 2048 bytes or greater. To avoid fragmentation, the maximum transmission unit (MTU) at the network layer should not exceed 1500, unless a peer MRU of 2048 or greater is specifically negotiated.

2.6 X.25

This topic describes the encapsulation of IP over X.25 networks, in accordance with ISO/IEC and CCITT standards. IP over X.25 networks is documented by RFC 1356 (which obsoletes RFC 877). RFC 1356 is a Draft Standard with a status of elective. The substantive change to the IP encapsulation over X.25 is an increase in the IP datagram MTU size, the X.25 maximum data packet size, the virtual circuit management, and the interoperable encapsulation over X.25 of protocols other than IP between multiprotocol routers and bridges.

One or more X.25 virtual circuits are opened on demand when datagrams arrive at the network interface for transmission. Protocol data units (PDUs) are sent as X.25 *complete packet sequences*. That is, PDUs begin on X.25 data packet boundaries and the M bit (more data) is used to fragment PDUs that are larger than one X.25 data packet in length. In the IP encapsulation, the PDU is the IP datagram. The first octet in the call user data (CUD) field (the first data octet in the Call Request packet) is used for protocol demultiplexing in accordance with the Subsequent Protocol Identifier (SPI) in ISO/IEC TR 9577. This field contains a one octet network-layer protocol identifier (NLPID), which identifies the network-layer protocol encapsulated over the X.25 virtual circuit. For the Internet community, the NLPID has four relevant values:

- The value hex CC (binary 11001100, decimal 204) is IP.

[1] CCITT Recommendations I.465 and V.120, “Data Terminal Equipment Communications over the Telephone Network with Provision for Statistical Multiplexing”, *CCITT Blue Book*, Volume VIII, Fascicle VIII.1, 1988

- The value hex 81 (binary 10000001, decimal 129) identifies ISO/IEC 8473 (CLNP).
- The value hex 82 (binary 10000010, decimal 130) is used specifically for ISO/IEC 9542 (ES-IS). If there is already a circuit open to carry CLNP, it is not necessary to open a second circuit to carry ES-IS.
- The value hex 80 (binary 10000000, decimal 128) identifies the use of the IEEE Subnetwork Access Protocol (SNAP) to further encapsulate and identify a single network-layer protocol. The SNAP-encapsulated protocol is identified by including a five-octet SNAP header in the Call Request CUD field immediately following the hex 80 octet. SNAP headers are not included in the subsequent X.25 data packets. Only one SNAP-encapsulated protocol can be carried over a virtual circuit opened using this encoding.

The value hex 00 identifies the null encapsulation used to multiplex multiple network-layer protocols over the same circuit. RFC 3232 contains one other non-CCITT and non-ISO/IEC value that has been used for Internet X.25 encapsulation identification, namely hex C5 (binary 11000101, decimal 197) for Blacker X.25. This value may continue to be used, but only by prior preconfiguration of the sending and receiving X.25 interfaces to support this value. The hex CD (binary 11001101, decimal 205), listed in RFC 3232 for ISO-IP, is also used by Blacker and can only be used by prior preconfiguration of the sending and receiving X.25 interfaces.

Each system must only accept calls for protocols it can process. Every Internet system must be able to accept the CC encapsulation for IP datagrams. Systems that support NLPIDs other than hex CC (for IP) should allow their use to be configured on a per-peer address basis. The Null encapsulation, identified by a NLPID encoding of hex 00, is used in order to multiplex multiple network-layer protocols over one circuit. When the Null encapsulation is used, each X.25 complete packet sequence sent on the circuit begins with a one-octet NLPID, which identifies the network-layer protocol data unit contained only in that particular complete packet sequence. Further, if the SNAP NLPID (hex 80) is used, the NLPID octet is immediately followed by the five-octet SNAP header, which is then immediately followed by the encapsulated PDU. The encapsulated network-layer protocol can differ from one complete packet sequence to the next over the same circuit.

Use of the single network-layer protocol circuits is more efficient in terms of bandwidth if only a limited number of protocols are supported by a system. It also allows each system to determine exactly which protocols are supported by its communicating partner. Other advantages include being able to use X.25 accounting to detail each protocol and different quality of service or flow control windows for different protocols. The Null encapsulation, for multiplexing, is useful when a system, for any reason (such as implementation restrictions or network

cost considerations), can only open a limited number of virtual circuits simultaneously. This is the method most likely to be used by a multiprotocol router to avoid using an unreasonable number of virtual circuits. If performing IEEE 802.1d bridging across X.25 is required, the Null encapsulation must be used.

IP datagrams must, by default, be encapsulated on a virtual circuit opened with the CC CUD. Implementations can also support up to three other possible encapsulations of IP:

- IP datagrams can be contained in multiplexed data packets on a circuit using the Null encapsulation. Such data packets are identified by a NLPID of hex CC.
- IP can be encapsulated within the SNAP encapsulation on a circuit. This encapsulation is identified by containing, in the 5-octet SNAP header, an Organizationally Unique Identifier (OUI) of hex 00-00-00 and Protocol Identifier (PID) of hex 08-00.
- On a circuit using the Null encapsulation, IP can be contained within the SNAP encapsulation of IP in multiplexed data packets.

2.7 Frame relay

The frame relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same frame relay network. The resulting set of interconnected devices forms a private frame relay group, which can be either fully interconnected with a complete *mesh* of virtual circuits, or only partially interconnected. In either case, each virtual circuit is uniquely identified at each frame relay interface by a data link connection identifier (DLCI). In most circumstances, DLCIs have strictly local significance at each frame relay interface. Frame relay is documented in RFC 2427, and is expanded in RFC 2590 to allow the transmission of IPv6 packets.

2.7.1 Frame format

All protocols must encapsulate their packets within a Q.922 Annex A frame[2]. Additionally, frames contain the necessary information to identify the protocol carried within the protocol data unit (PDU), thus allowing the receiver to properly

[2] International Telecommunication Union, "ISDN Data Link Layer Specification for Frame Mode Bearer Services," ITU-T Recommendation Q.922, 1992

process the incoming packet (refer to Figure 2-6 on page 43). The format will be as follows:

- The control field is the Q.922 control field. The UI (0x03) value is used unless it is negotiated otherwise. The use of XID (0xAF or 0xBF) is permitted.
- The pad field is used to align the data portion (beyond the encapsulation header) of the frame to a two octet boundary. If present, the pad is a single octet and must have a value of zero.
- The Network Level Protocol ID (NLPID) field is administered by ISO and the ITU. It contains values for many different protocols, including IP, CLNP, and IEEE Subnetwork Access Protocol (SNAP). This field tells the receiver what encapsulation or what protocol follows. Values for this field are defined in ISO/IEC TR 9577[3]. An NLPID value of 0x00 is defined within ISO/IEC TR 9577 as the null network layer or inactive set. Because it cannot be distinguished from a pad field, and because it has no significance within the context of this encapsulation scheme, an NLPID value of 0x00 is invalid under the frame relay encapsulation.

There is no commonly implemented minimum or maximum frame size for frame relay. A network must, however, support at least a 262-octet maximum. Generally, the maximum will be greater than or equal to 1600 octets, but each frame relay provider will specify an appropriate value for its network. A frame relay data terminal equipment (DTE) must allow the maximum acceptable frame size to be configurable.

[3] "Information technology -- Protocol identification in the network layer," ISO/IEC TR 9577, 1999

Figure 2-6 shows the format for a frame relay packet.

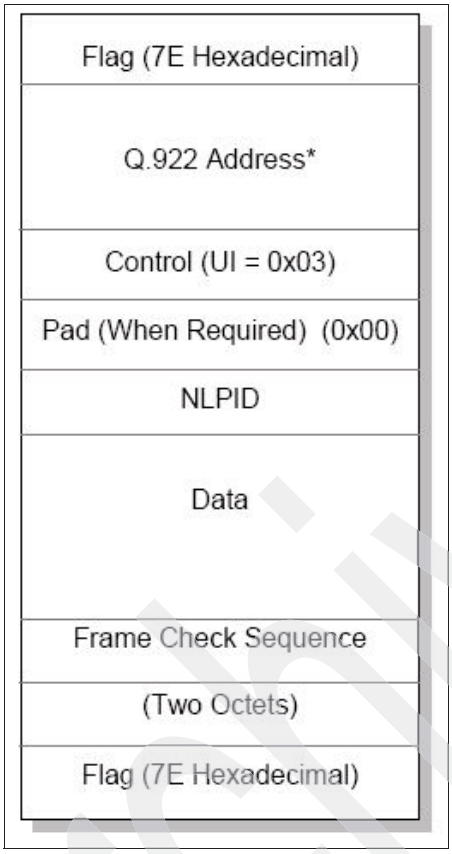


Figure 2-6 Frame relay packet format

2.7.2 Interconnect issues

There are two basic types of data packets that travel within the frame relay network: routed packets and bridged packets. These packets have distinct formats and must contain an indicator that the destination can use to correctly interpret the contents of the frame. This indicator is embedded within the NLPID and SNAP header information.

2.7.3 Data link layer parameter negotiation

Frame relay stations may choose to support the Exchange Identification (XID) specified in Appendix III of Q.922[4]. This XID exchange allows the following parameters to be negotiated at the initialization of a frame relay circuit: maximum

Archived

frame size, retransmission timer, and the maximum number of outstanding information (I) frames.

If this exchange is not used, these values must be statically configured by mutual agreement of data link connection (DLC) endpoints, or must be defaulted to the values specified in Section 5.9 of Q.922[4].

There are situations in which a frame relay station might want to dynamically resolve a protocol address over permanent virtual circuits (PVCs). This can be accomplished using the standard Address Resolution Protocol (ARP) encapsulated within a SNAP-encoded frame relay packet.

Because of the inefficiencies of emulating broadcasting in a frame relay environment, a new address resolution variation was developed. It is called Inverse ARP, and describes a method for resolving a protocol address when the hardware address is already known. In a frame relay network, the known hardware address is the DLCI. Support for Inverse ARP is not required to implement this specification, but it has proven useful for frame relay interface autoconfiguration.

Stations must be able to map more than one IP address in the same IP subnet to a particular DLCI on a frame relay interface. This need arises from applications such as remote access, where servers must act as ARP proxies for many dial-in clients, each assigned a unique IP address while sharing bandwidth on the same DLC. The dynamic nature of such applications results in frequent address association changes with no effect on the DLC's status.

As with any other interface that uses ARP, stations can learn the associations between IP addresses and DLCIs by processing unsolicited (*gratuitous*) ARP requests that arrive on the DLC. If one station wants to inform its peer station on the other end of a frame relay DLC of a new association between an IP address and that PVC, it should send an unsolicited ARP request with the source IP address equal to the destination IP address, and both set to the new IP address being used on the DLC. This allows a station to “announce” new client connections on a particular DLCI. The receiving station must store the new association, and remove any old existing association, if necessary, from any other DLCI on the interface.

2.7.4 IP over frame relay

Internet Protocol (IP) datagrams sent over a frame relay network conform to the encapsulation described previously. Within this context, IP can be encapsulated

[4] International Telecommunication Union, “ISDN Data Link Layer Specification for Frame Mode Bearer Services,” ITU-T Recommendation Q.922, 1992

in two different ways: NLPID value, indicating IP, or NLPID value, indicating SNAP.

Although both of these encapsulations are supported under the given definitions, it is advantageous to select only one method as the appropriate mechanism for encapsulating IP data. Therefore, encapsulate IP data using the NLPID value of 0xcc, indicating an IP packet. This option is more efficient, because it transmits 48 fewer bits without the SNAP header and is consistent with the encapsulation of IP in an X.25 network.

2.8 PPP over SONET and SDH circuits

This discussion describes the use of the PPP encapsulation over Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) links, which is documented by RFC 2615. Because SONET and SDH are, by definition, point-to-point circuits, PPP is well suited for use over these links. SONET is an octet-synchronous multiplex scheme that defines a family of standard rates and formats. Despite the name, it is not limited to optical links. Electrical specifications have been defined for single-mode fiber, multimode fiber, and CATV 75 ohm coaxial cable. The transmission rates are integral multiples of 51.840 Mbps, which can be used to carry T3/E3 bit-synchronous signals. The allowed multiples are currently specified as shown in Table 2-1. Additionally, the CCITT Synchronous Digital Hierarchy defines a subset of SONET transmission rates beginning at 155.52 Mbps.

Table 2-1 SONET speed hierarchy

Kind	Length	Meaning
0	-	End of option list
1	-	No-Operation
2	4	Maximum segment size
3	3	Window scale
4	2	Sack-Permitted
5	X	Sack
8	10	Time stamps

2.8.1 Physical layer

PPP presents an octet interface to the physical layer. There is no provision for sub-octets to be supplied or accepted. SONET and SDH links are full-duplex by definition. The octet stream is mapped into the SONET/SDH Synchronous Payload Envelope (SPE) with the octet boundaries aligned with the SPE octet boundaries. No scrambling is needed during insertion into the SPE. The path signal label is intended to indicate the contents of the SPE. The experimental value of 207 (hex CF) is used to indicate PPP. The multiframe indicator is currently unused and must be zero.

The basic rate for PPP over SONET/SDH is that of STS-3c/STM-1 at 155.52 Mbps. The available information bandwidth is 149.76 Mbps, which is the STS-3c/STM-1 SPE with section, line, and path inefficiencies removed. This is the same upper layer mapping that is used for ATM and FDDI. Lower signal rates must use the Virtual Tributary (VT) mechanism of SONET/SDH. This maps existing signals up to T3/E3 rates asynchronously into the SPE or uses available clocks for bit-synchronous and byte-synchronous mapping. Higher signal rates should conform to the SDH STM series rather than the SONET STS series as equipment becomes available. The STM series progresses in powers of 4 instead of 3 and employs fewer steps, which is likely to simplify multiplexing and integration.

2.9 Multi-Path Channel+ (MPC+)

The MPC support is a protocol layer that allows multiple read and write subchannels to be treated as a single transmission group between the host and channel-attached devices. One level of MPC support, high performance data transfer (HPDT), also referred to as MPC+, provides more efficient transfer of data than non-HPDT MPC connections. Multi-Path Channel+ (MPC+) connections enable you to define a single transmission group (TG) that uses multiple write-direction and read-direction subchannels. Because each subchannel operates in only one direction, the half-duplex turnaround time that occurs with other channel-to-channel connections is reduced.

If at least one read and one write path is allocated successfully, the MPC+ channel connection is activated. Additional paths (defined but not online) in an MPC+ group can later be dynamically added to the active group using the MVS™ VARY device ONLINE command.

For example, if there is a need for an increase in capacity to allow for extra traffic over a channel, additional paths can be added to the active group without disruption. Similarly, paths can be deleted from the active group when no longer needed using the MVS VARY device OFFLINE command.

2.10 Asynchronous transfer mode (ATM)

ATM-based networks are of increasing interest for both local and wide area applications. The ATM architecture is different from the standard LAN architectures and, for this reason, changes are required so that traditional LAN products will work in the ATM environment. In the case of TCP/IP, the main change required is in the network interface to provide support for ATM.

There are several approaches already available, two of which are important to the transport of TCP/IP traffic. We describe these in 2.10.2, “Classical IP over ATM” on page 50 and 2.10.3, “ATM LAN emulation” on page 56. We also compare them in 2.10.4, “Classical IP over ATM versus LAN emulation” on page 59.

2.10.1 Address resolution (ATMARP and InATMARP)

The address resolution in an ATM logical IP subnet is done by the ATM Address Resolution Protocol (ATMARP), based on RFC 826 (also STD 37), and the Inverse ATM Address Resolution Protocol (InATMARP), based on RFC 2390. ATMARP is the same protocol as the ARP protocol, with extensions needed to support ARP in a unicast server ATM environment. InATMARP is the same protocol as the original InARP protocol, but applied to ATM networks. Use of these protocols differs depending on whether permanent virtual connections (PVCs) or switched virtual connections (SVCs) are used.

Both ATMARP and InATMARP are defined in RFC 2225; a proposed standard with a state of elective. We describe the encapsulation of ATMARP and InATMARP requests/replies in 2.10.2, “Classical IP over ATM” on page 50.

InATMARP

The ARP protocol resolves a host's hardware address for a known IP address. The InATMARP protocol resolves a host's IP address for a known hardware address. In a switched environment, you first establish a virtual connection (VC) of either a permanent virtual connection (PVC) or switched virtual connection (SVC) in order to communicate with another station. Therefore, you know the exact hardware address of the partner by administration, but the IP address is unknown. InATMARP provides dynamic address resolution. InARP uses the same frame format as the standard ARP, but defines two new operation codes:

- InARP request=8
- InARP reply=9

See “ARP packet generation” on page 120 for more details.

Basic InATMARP operates essentially the same as ARP, with the exception that InATMARP does not broadcast requests. This is because the hardware address of the destination station is already known. A requesting station simply formats a request by inserting its source hardware and IP address and the known target hardware address. It then zero fills the target protocol address field and sends it directly to the target station. For every InATMARP request, the receiving station formats a reply using the source address from the request as the target address of the reply. Both sides update their ARP tables. The hardware type value for ATM is 19 decimal and the EtherType field is set to 0x806, which indicates ARP, according to RFC 3232.

Address resolution in a PVC environment

In a PVC environment, each station uses the InATMARP protocol to determine the IP addresses of all other connected stations. The resolution is done for those PVCs that are configured for LLC/SNAP encapsulation. It is the responsibility of each IP station supporting PVCs to revalidate ARP table entries as part of the aging process.

Address resolution in an SVC environment

SVCs require support for ATMARP in the non-broadcast environment of ATM. To meet this need, a single ATMARP server must be located within the Logical IP Subnetwork (LIS) (see “The Logical IP Subnetwork (LIS)” on page 53). This server has authoritative responsibility for resolving the ATMARP requests of all IP members within the LIS. For an explanation of ATM terms, refer to 2.10.2, “Classical IP over ATM” on page 50.

The server itself does not actively establish connections. It depends on the clients in the LIS to initiate the ATMARP registration procedure. An individual client connects to the ATMARP server using a point-to-point VC. The server, upon the completion of an ATM call/connection of a new VC specifying LLC/SNAP encapsulation, will transmit an InATMARP request to determine the IP address of the client. The InATMARP reply from the client contains the information necessary for the ATMARP server to build its ATMARP table cache. This table consists of:

- IP address
- ATM address
- Time stamp
- Associated VC

This information is used to generate replies to the ATMARP requests it receives.

Note: The ATMARP server mechanism requires that each client be administratively configured with the ATM address of the ATMARP server.

ARP table add/update algorithm

Consider the following points:

- If the ATMARP server receives a new IP address in an InATMARP reply, the IP address is added to the ATMARP table.
- If the InATMARP IP address duplicates a table entry IP address and the InATMARP ATM address does not match the table entry ATM address, and there is an open VC associated with that table entry, the InATMARP information is discarded and no modifications to the table are made.
- When the server receives an ATMARP request over a VC, where the source IP and ATM address match the association already in the ATMARP table and the ATM address matches that associated with the VC, the server updates the timeout on the source ATMARP table entry. For example, if the client is sending ATMARP requests to the server over the same VC that it used to register its ATMARP entry, the server notes that the client is still “alive” and updates the timeout on the client's ATMARP table entry.
- When the server receives an ARP_REQUEST over a VC, it examines the source information. If there is no IP address associated with the VC over which the ATMARP request was received and if the source IP address is not associated with any other connection, the server adds this station to its ATMARP table. This is not the normal way because, as mentioned earlier, it is the responsibility of the client to register at the ATMARP server.

ATMARP table aging

ATMARP table entries are valid:

- In clients for a maximum time of 15 minutes
- In servers for a minimum time of 20 minutes

Prior to aging an ATMARP table entry, the ATMARP server generates an InARP_REQUEST on any open VC associated with that entry and decides what to do according to the following rules:

- If an InARP_REPLY is received, that table entry is updated and not deleted.
- If there is no open VC associated with the table entry, the entry is deleted.

Therefore, if the client does not maintain an open VC to the server, the client must refresh its ATMARP information with the server at least once every 20 minutes. This is done by opening a VC to the server and exchanging the initial InATMARP packets.

The client handles the table updates according to the following:

- When an ATMARP table entry ages, the ATMARP client invalidates this table entry.
- If there is no open VC associated with the invalidated entry, that entry is deleted.
- In the case of an invalidated entry and an open VC, the ATMARP client revalidates the entry prior to transmitting any non-address resolution traffic on that VC. There are two possibilities:
  - In the case of a PVC, the client validates the entry by transmitting an InARP_REQUEST and updating the entry on receipt of an InARP_REPLY.
  - In the case of an SVC, the client validates the entry by transmitting an ARP_REQUEST to the ATMARP server and updating the entry on receipt of an ARP_REPLY.
- If a VC with an associated invalidated ATMARP table entry is closed, that table entry is removed.

As mentioned earlier, every ATM IP client that uses SVCs must know its ATMARP server's ATM address for the particular LIS. This address must be named at every client during customization. There is at present no well-known ATMARP server address defined.

2.10.2 Classical IP over ATM

The definitions for implementations of classical IP over asynchronous transfer mode (ATM) are described in RFC 2225, which is a proposed standard with a status of elective. This RFC considers only the application of ATM as a direct replacement for the “wires” and local LAN segments connecting IP endstations (*members*) and routers operating in the classical LAN-based paradigm. Issues raised by MAC level bridging and LAN emulation are not covered. Additionally, IP over ATM was expanded by RFC 2492, which defines the transmission of IPv6 over ATM.

For ATM Forum's method of providing ATM migration, see 2.10.3, “ATM LAN emulation” on page 56.

Initial deployment of ATM provides a LAN segment replacement for:

- Ethernets, token rings, or FDDI networks
- Local area backbones between existing (non-ATM) LANs
- Dedicated circuits of frame relay PVCs between IP routers

RFC 2225 also describes extensions to the ARP protocol (RFC 826) in order to work over ATM. We discuss this separately in 2.10.1, “Address resolution (ATMARP and InATMARP)” on page 47.

First, some ATM basics:

Cells All information (voice, image, video, data, and so on) is transported through the network in very short (48 data bytes plus a 5-byte header) blocks called *cells*.

Routing Information flow is along paths (called *virtual channels*) set up as a series of pointers through the network. The cell header contains an identifier that links the cell to the correct path that it will take toward its destination.

Cells on a particular virtual channel always follow the same path through the network and are delivered to the destination in the same order in which they were received.

Hardware-based switching
ATM is designed such that simple hardware-based logic elements can be employed at each node to perform the switching. On a link of 1 Gbps, a new cell arrives and a cell is transmitted every .43 microseconds. There is not a lot of time to decide what to do with an arriving packet.

Virtual Connection (VC)
ATM provides a virtual connection switched environment. VC setup can be done on either a permanent virtual connection (PVC) or a dynamic switched virtual connection (SVC) basis. SVC call management is performed by implementations of the Q.93B protocol.

End-user interface The only way for a higher layer protocol to communicate across an ATM network is over the ATM Adaptation Layer (AAL). The function of this layer is to perform the mapping of protocol data units (PDUs) into the information field of the ATM cell and vice versa. There are four different AAL types defined: AAL1, AAL2, AAL3/4, and AAL5. These AALs offer different services for higher layer protocols. Here are the characteristics of AAL5, which is used for TCP/IP:

- Message mode and streaming mode
- Assured delivery
- Non-assured delivery (used by TCP/IP)
- Blocking and segmentation of data
- Multipoint operation

AAL5 provides the same functions as a LAN at the Medium Access Control (MAC) layer. The AAL type is known by the VC endpoints through the cell setup mechanism and is not carried in the ATM cell header. For PVCs, the AAL type is administratively configured at the endpoints when the connection (circuit) is set up. For SVCs, the AAL type is communicated along the VC path through Q.93B as part of call setup establishment and the endpoints use the signaled information for configuration. ATM switches generally do not care about the AAL type of VCs. The AAL5 format specifies a packet format with a maximum size of 64 KB - 1 byte of user data. The *primitives*, which the higher layer protocol has to use in order to interface with the AAL layer (at the AAL service access point, or SAP), are rigorously defined. When a high-layer protocol sends data, that data is processed first by the adaptation layer, then by the ATM layer, and then the physical layer takes over to send the data to the ATM network. The cells are transported by the network and then received on the other side first by the physical layer, then processed by the ATM layer, and then by the receiving AAL. When all this is complete, the information (data) is passed to the receiving higher layer protocol. The total function performed by the ATM network has been the non-assured transport (it might have lost some) of information from one side to the other. Looked at from a traditional data processing viewpoint, all the ATM network has done is to replace a physical link connection with another kind of physical connection. All the higher layer network functions must still be performed (for example, IEEE 802.2).

Addressing An ATM Forum endpoint address is either encoded as a 20-byte OSI NSAP-based address (used for private network addressing, three formats possible) or is an E.164 Public UNI address (telephone number style address used for public ATM networks).[5]

Broadcast, multicast There are currently no broadcast functions similar to LANs provided. But there is a multicast function available. The ATM term for multicast is *point-to-multipoint connection*.

[5] The ATM Forum is a worldwide organization, aimed at promoting ATM within the industry and the user community. The membership includes more than 500 companies representing all sectors of the communications and computer industries, as well as a number of government agencies, research organizations, and users.

The Logical IP Subnetwork (LIS)

The term LIS was introduced to map the logical IP structure to the ATM network. In the LIS scenario, each separate administrative entity configures its hosts and routers within a closed logical IP subnetwork (same IP network/subnet number and address mask). Each LIS operates and communicates independently of other LISs on the same ATM network. Hosts that are connected to an ATM network communicate directly to other hosts within the same LIS. This implies that all members of a LIS are able to communicate through ATM with all other members in the same LIS. (VC topology is fully meshed.) Communication to hosts outside of the local LIS is provided through an IP router. This router is an ATM endpoint attached to the ATM network that is configured as a member of one or more LISs. This configuration might result in a number of separate LISs operating over the same ATM network. Hosts of differing IP subnets must communicate through an intermediate IP router, even though it might be possible to open a direct VC between the two IP members over the ATM network.

Multiprotocol encapsulation

If you want to use more than one type of network protocol (IP, IPX™, and so on) concurrently over a physical network, you need a method of multiplexing the different protocols. This can be done in the case of ATM either by VC-based multiplexing or LLC encapsulation. If you choose VC-based multiplexing, you have to have a VC for each different protocol between the two hosts. The LLC encapsulation provides the multiplexing function at the LLC layer and therefore needs only one VC. TCP/IP uses, according to RFC 2225 and 2684, the second method, because this kind of multiplexing was already defined in RFC 1042 for all other LAN types, such as Ethernet, token ring, and FDDI. With this definition, IP uses ATM simply as a LAN replacement. All the other benefits ATM has to offer, such as transportation of isochronous traffic, and so on, are not used. There is an IETF working group with the mission of improving the IP implementation and to interface with the ATM Forum in order to represent the interests of the Internet community for future standards.

Archive

To be exact, the TCP/IP PDU is encapsulated in an IEEE 802.2 LLC header followed by an IEEE 802.1a SubNetwork Attachment Point (SNAP) header and carried within the payload field of an AAL5 CPCS-PDU (Common Part Convergence Sublayer). The following figure shows the AAL5 CPCS-PDU format (Figure 2-7).

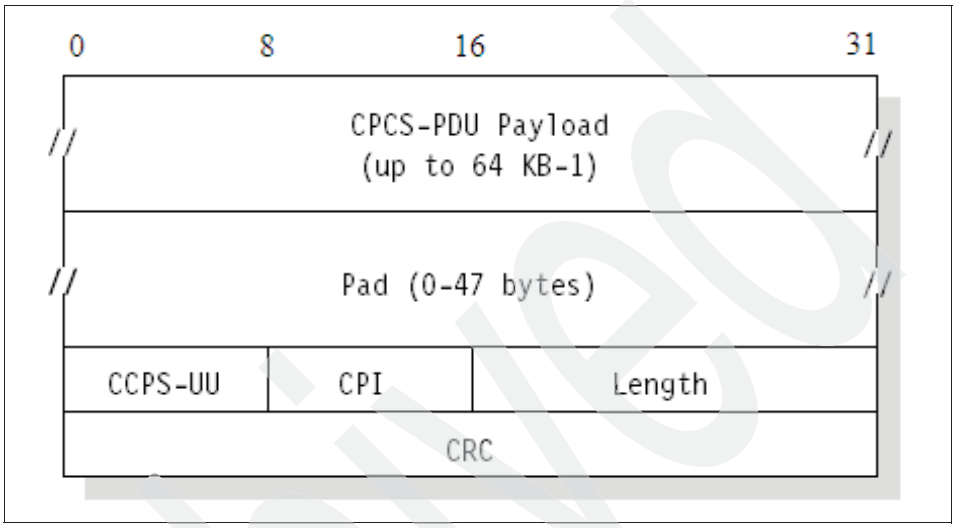


Figure 2-7 AAL5 CPCS-PDU format

Where:

CPCS-PDU Payload The CPCS-PDU payload is shown in Figure 2-8 on page 55.

Pad The Pad field pads out the CDCS-PDU to fit exactly into the ATM cells.

CPCS-UU The CPCS-UU (User-to-User identification) field is used to transparently transfer CPCS user-to-user information. This field has no function for the encapsulation and can be set to any value.

CPI The Common Part Indicator (CPI) field aligns the CPCS-PDU trailer with 64 bits.

Length The Length field indicates the length, in bytes, of the payload field. The maximum value is 65535, which is 64 KB - 1.

CRC The CRC field protects the entire CPCS-PDU, except the CRC field itself.

The following figure shows the payload format for routed IP PDUs (Figure 2-8).

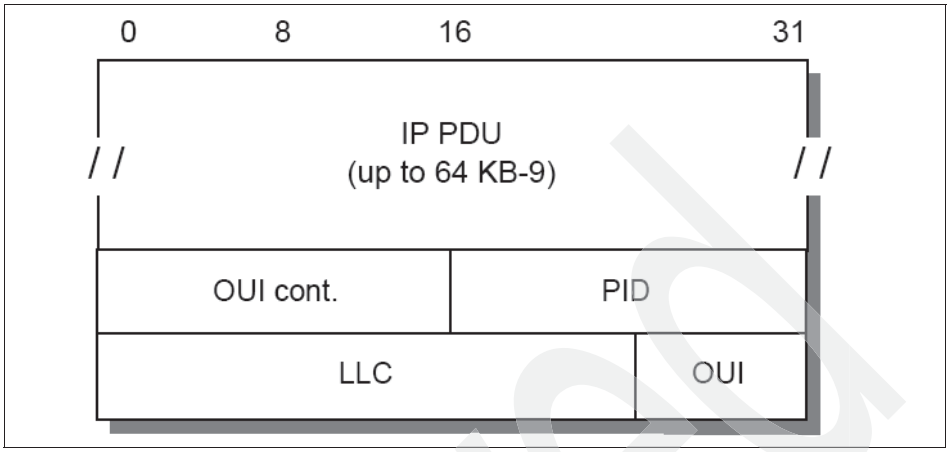


Figure 2-8 CPCS-PDU payload format for IP PDUs

Where:

IP PDU Normal IP datagram, starting with the IP header.

LLC A 3-byte LLC header with the format DSAP-SSAP-Ctrl. For IP data, it is set to 0xAA-AA-03 to indicate the presence of a SNAP header. The Ctrl field always has the value 0x03, specifying Unnumbered Information Command PDU.

OUI The 3-byte Organizationally Unique Identifier (OUI) identifies an organization that administers the meaning of the following 2-byte Protocol Identifier (PID). To specify an EtherType in PID, the OUI has to be set to 0x00-00-00.

PID The Protocol Identifier (PID) field specifies the protocol type of the following PDU. For IP datagrams, the assigned EtherType or PID is 0x08-00.

The default MTU size for IP members in an ATM network is discussed in RFC 2225 and defined to be 9180 bytes. The LLC/SNAP header is 8 bytes; therefore, the default ATM AAL5 PDU size is 9188 bytes. The possible values can be between zero and 65535. You are allowed to change the MTU size, but then all members of a LIS must be changed as well in order to have the same value. RFC 1755 recommends that all implementations should support MTU sizes up to and including 64 KB.

Archived

The address resolution in an ATM network is defined as an extension of the ARP protocol and is described in 2.10.1, “Address resolution (ATMARP and InATMARP)” on page 47.

There is no mapping from IP broadcast or multicast addresses to ATM *broadcast* or multicast addresses available. But there are no restrictions for transmitting or receiving IP datagrams specifying any of the four standard IP broadcast address forms as described in RFC 1122. Members, upon receiving an IP broadcast or IP subnet broadcast for their LIS, must process the packet as though addressed to that station.

2.10.3 ATM LAN emulation

Another approach to provide a migration path to a native ATM network is ATM LAN emulation. ATM LAN emulation is still under construction by ATM Forum working groups. For the IETF approach, see 2.10.2, “Classical IP over ATM” on page 50. There is no ATM Forum implementation agreement available covering virtual LANs over ATM, but there are some basic agreements on the different proposals made to the ATM Forum. The following descriptions are based on the IBM proposals.

The concept of ATM LAN emulation is to construct a system such that the workstation application software “thinks” it is a member of a real shared medium LAN, such as a token ring. This method maximizes the reuse of existing LAN software and significantly reduces the cost of migration to ATM. In PC LAN environments, for example, the LAN emulation layer can be implemented under the NDIS/ODI-type interface. With such an implementation, all the higher layer protocols, such as IP, IPX, NetBIOS, and SNA, can be run over ATM networks without any change.

Archive

Refer to Figure 2-9 for the implementation of token ring and Ethernet.

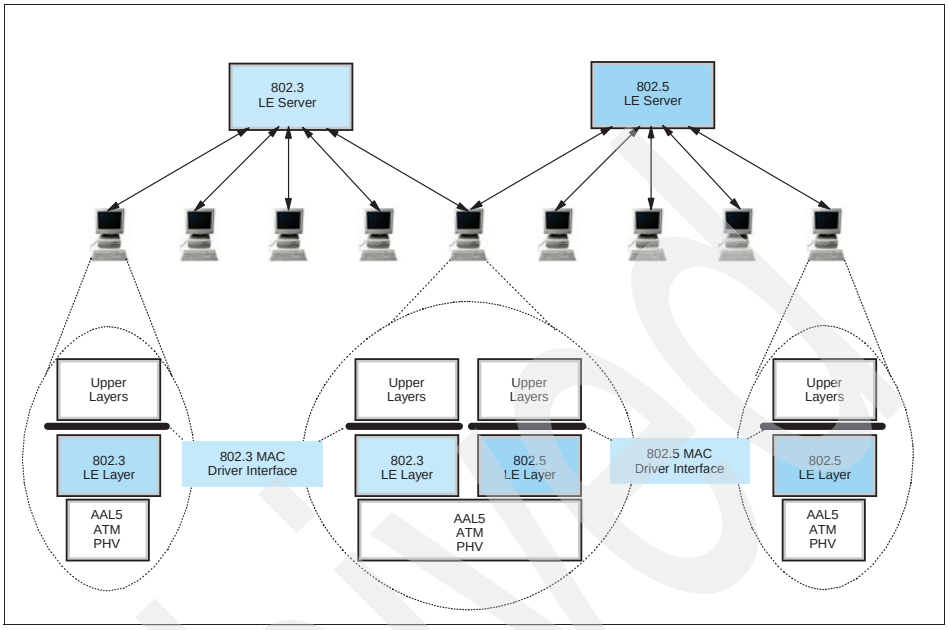


Figure 2-9 Ethernet and token-ring LAN emulation

LAN emulation layer (workstation software)

Each workstation that performs the LE function needs to have software to provide the LE service. This software is called the LAN emulation layer (LE layer). It provides the interface to existing protocol support (such as IP, IPX, IEEE 802.2 LLC, NetBIOS, and so on) and emulates the functions of a real, shared medium LAN. This means that no changes are needed to existing LAN application software to use ATM services. The LE layer interfaces to the ATM network through a hardware ATM adapter.

The primary function of the LE layer is to transfer encapsulated LAN frames (arriving from higher layers) to their destination either directly (over a direct VC) or through the LE server. This is done by using AAL5 services provided by ATM.

Each LE layer has one or more LAN addresses as well as an ATM address.

A separate instance (logical copy or LE client) of the LE layer is needed in each workstation for each different LAN or type of LAN to be supported. For example, if both token-ring and Ethernet LAN types are to be emulated, you need two LE layers. In fact, they will probably just be different threads within the same copy of the same code, but they are logically separate LE layers. Use separate LE layers also if one workstation needs to be part of two different emulated token-ring

LANs. Each separate LE layer needs a different MAC address, but can share the same physical ATM connection (adapter).

LAN emulation server

The basic function of the LE server is to provide directory, multicast, and address resolution services to the LE layers in the workstations. It also provides a connectionless data transfer service to the LE layers in the workstations, if needed.

Each emulated LAN must have an LE server. It would be possible to have multiple LE servers sharing the same hardware and code (via multithreading), but the LE servers are logically separate entities. As for the LE layers, an emulated token-ring LAN cannot have members that are emulating an Ethernet LAN. Thus, an instance of an LE server is dedicated to a single type of LAN emulation. The LE server can be physically internal to the ATM network or provided in an external device, but logically it is always an external function that simply uses the services provided by ATM to do its job.

Default VCs

A default VC is a connection between an LE layer in a workstation and the LE server. These connections can be permanent or switched.

All LE control messages are carried between the LE layer and the LE server on the default VC. Encapsulated data frames can also be sent on the default VC.

The presence of the LE server and the default VCs is necessary for the LE function to be performed.

Direct VCs

Direct VCs are connections between LE layers in the end systems. They are always switched and set up on demand. If the ATM network does not support switched connections, you cannot have direct VCs, and all the data must be sent through the LE server on default VCs. If there is no direct VC available for any reason, data transfer must take place through the LE server. (There is no other way.)

Direct VCs are set up on request by an LE layer. (The server cannot set them up, because there is no third-party call setup function in ATM.) The ATM address of a destination LE layer is provided to a requesting LE layer by the LE server. Direct VCs stay in place until one of the partner LE layers decides to end the connection (because there is no more data).

Initialization

During initialization, the LE layer (workstation) establishes the default VC with the LE server. It also discovers its own ATM address, which is needed if it is to later set up direct VCs.

Registration

In this phase, the LE layer (workstation) registers its MAC addresses with the LE server. Other things, such as filtering requirements (optional), can be provided.

Management and resolution

This is the method used by ATM endstations to set up direct VCs with other endstations (LE layers). This function includes mechanisms for learning the ATM address of a target station, mapping the MAC address to an ATM address, storing the mapping in a table, and managing the table.

For the server, this function provides the means for supporting the use of direct VCs by endstations. This includes a mechanism for mapping the MAC address of an end system to its ATM address, storing the information, and providing it to a requesting endstation.

This structure maintains full LAN function and can support most higher layer LAN protocols. Reliance on the server for data transfer is minimized by using switched VCs for the transport of most bulk data.

2.10.4 Classical IP over ATM versus LAN emulation

These two approaches to providing an easier way to migrate to ATM were made with different goals in mind.

Classical IP over ATM defines an encapsulation and address resolution method. The definitions are made for IP only and not for use with other protocols. So if you have applications requiring other protocol stacks (such as IPX or SNA), IP over ATM will not provide a complete solution. However, if you only have TCP or UDP-based applications, this might be the better solution, because this specialized adaptation of the IP protocol to the ATM architecture is likely to produce fewer inefficiencies than a more global solution. Another advantage of this implementation is the use of some ATM-specific functions, such as large MTU sizes.

The major goal of the ATM Forum's approach is to run layer 3 and higher protocols unchanged over the ATM network. This means that existing protocols, for example, TCP/IP, IPX, NetBIOS, and SNA, and their applications can use the benefits of the fast ATM network without any changes. The mapping for all protocols is already defined. The LAN emulation (LANE) layer provides all the

services of a classic LAN; thus, the upper layer does not know of the existence of ATM. This is both an advantage and a disadvantage, because the knowledge of the underlying network could be used to provide a more effective implementation.

In the near future, both approaches will be used depending on the particular requirements. Over time, when the mapping of applications to ATM is fully defined and implemented, the scheme of a dedicated ATM implementation might be used.

2.11 Multiprotocol over ATM (MPOA)

The objectives of MPOA are to:

- Provide end-to-end layer 3 internetworking connectivity across an ATM network. This is for hosts that are attached either:
  - Directly to the ATM network
  - Indirectly to the ATM network on an existing LAN
- Support the distribution of the internetwork layer (for example, an IP subnet) across traditional and ATM-attached devices. Removes the port to layer 3 network restriction of routers to enable the building of protocol-based virtual LANs (VLANs).
- Ensure interoperability among the distributed routing components while allowing flexibility in implementations.
- Address how internetwork-layer protocols use the services of an ATM network.

Although the name is multiprotocol over ATM, the actual work being done at the moment in the MPOA subworking group is entirely focused on IP.

2.11.1 Benefits of MPOA

MPOA represents the transition from LAN emulation to direct exploitation of ATM by the internetwork-layer protocols. The advantages are:

- Protocols see ATM as more than just another link. Therefore, we are able to exploit the facilities of ATM.
- Increases efficiency of the traditional LAN frame structure.

Archived

The MPOA solution has the following benefits over both Classical IP (RFC 2225) and LAN emulation solutions:

- Lower latency by allowing direct connectivity between end systems that can cut across subnet boundaries. This is achieved by minimizing the need for multiple hops through ATM routers for communication between end systems on different virtual LANs.
- Higher aggregate layer 3 forwarding capacity by distributing processing functions to the edge of the network.
- Allows mapping of specific flows to specific QoS characteristics.
- Allows a layer 3 subnet to be distributed across a physical network.

2.11.2 MPOA logical components

The MPOA solution consists of a number of logical components and information flows between those components. The logical components are of two kinds:

MPOA server MPOA servers maintain *complete* knowledge of the MAC and internetworking layer topologies for the IASGs they serve. To accomplish this, they exchange information among themselves and with MPOA clients.

MPOA client MPOA clients maintain local caches of mappings (from packet prefix to ATM information). These caches are populated by requesting the information from the appropriate MPOA server on an as-needed basis.

The layer 3 addresses associated with an MPOA client represent either the layer 3 address of the client itself, or the layer 3 addresses reachable through the client. (The client has an edge device or router.)

An MPOA client will connect to its MPOA server to register the client's ATM address and the layer 3 addresses reachable by the client.

Archive

2.11.3 MPOA functional components

The mapping between the logical and physical components are split between the following layers:

- MPOA functional group layer
- LAN emulation layer
- Physical layer

The MPOA solution will be implemented into various functional groups that include:

- Internetwork Address Sub-Group (IASG): A range of internetwork layer addresses (for example, an IPv4 subnet). Therefore, if a host operates two internetwork-layer protocols, it will be a member of, at least, two IASGs.
- Edge Device Functional Group (EDFG): EDFG is the group of functions performed by a device that provides internetworking level connections between a traditional subnetwork and ATM.
  - An EDFG implements layer 3 packet forwarding, but does not execute any routing protocols (executed in the RSFG).
  - Two types of EDFG are allowed, *simple* and *smart*:
    - Smart EDFGs request resolution of internetwork addresses (that is, it will send a query ARP type frame if it does not have an entry for the destination).
    - Simple EDFGs will send a frame via a default class to a default destination if no entry exists.
  - A coresident proxy LEC function is required.
- ATM-Attached Host Functional Group (AHFG): AHFG is the group of functions performed by an ATM-attached host that is participating in the MPOA network.

  A coresident proxy LEC function is optional.

  Within an IASG, LAN emulation is used as a transport mechanism to either traditional devices or LAN emulation devices, in which case access to a LEC is required. If the AHFG will not be communicating with LANE or other devices, a co-resident LEC is not required.
- IASG Coordination Functional Group (ICFG): ICFG is the group of functions used to coordinate the distribution of a single IASG across multiple traditional LAN ports on one or more EDFG or ATM device, or both. The ICFG tracks the location of the functional components so that it is able to respond to queries for layer 3 addresses.

- Default Forwarder Function Group (DFFG): In the absence of direct client-to-client connectivity, the DFFG provides default forwarding for traffic destined either within or outside the IASG.
  - Provides internetwork layer multicast forwarding in an IASG; that is, the DFFG acts as the multicast server (MCS) in an MPOA-based MARS implementation.
  - Provides *proxy* LAN emulation function for AHFGs (that is, for AHFGs that do not have a LANE client) to enable AHFGs to send/receive traffic with earlier enterprise-attached systems.
- Route Server Functional Group (RSFG): RSFG performs internetworking level functions in an MPOA network. This includes:
  - Running conventional internetworking routing protocols (for example, OSPF, RIP, and BGP)
  - Providing address resolution between IASGs, handling requests, and building responses
- Remote Forwarder Functional Group (RFFG): RFFG is the group of functions performed in association with forwarding traffic from a source to a destination, where these can be either an IASG or an MPOA client. An RFFG is synonymous with the *default router* function of a typical IPv4 subnet.

Note: One or more of these functional groups can co-reside in the same physical entity. MPOA allows arbitrary physical locations of these groups.

2.11.4 MPOA operation

The MPOA system operates as a set of functional groups that exchange information in order to exhibit the desired behavior. To provide an overview of the MPOA system, the behavior of the components is described in a sequence order by *significant events*:

Configuration Ensures that all functional groups have the appropriate set of administrative information.

Registration and discovery Includes the functional groups informing each other of their existence and of the identities of attached devices and EDFGs informing the ICFG of earlier devices.

Destination resolution	The action of determining the route description given a destination internetwork layer address and possibly other information (for example, QoS). This is the part of the MPOA system that allows it to perform cut-through (with respect to IASG boundaries).
Data transfer	To get internetworking layer data from one MPOA client to another.
Intra-IASG coordination	The function that enables IASGs to be spread across multiple physical interfaces.
Routing protocol support	Enables the MPOA system to interact with traditional internetworks.
Spanning tree support	Enables the MPOA system to interact with existing extended LANs.
Replication Support	Provides for replication of key components for reasons of capacity or resilience.

2.12 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 826 – Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware (November 1982)
- RFC 894 – Standard for the Transmission of IP Datagrams over Ethernet Networks (April 1984)
- RFC 948 - Two Methods for the Transmission of IP Datagrams over IEEE 802.3 Networks (June 1985)
- RFC 1042 – Standard for the Transmission of IP Datagrams over IEEE 802 Networks (February 1988)
- RFC 1055 – Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP (June 1988)
- RFC 1144 – Compressing TCP/IP Headers for Low-Speed Serial Links (February 1990)
- RFC 1188 – Proposed Standard for the Transmission of IP Datagrams over FDDI Networks (October 1990)
- RFC 1329 – Thoughts on Address Resoluation for Dual MAC FDDI Networks (May 1992)

- RFC 1356 – Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode (August 1992)
- RFC 1390 – Transmission of IP and ARP over FDDI Networks (January 1993)
- RFC 1618 – PPP over ISDN (May 1994)
- RFC 1661 – The Point-to-Point Protocol (PPP) (July 1994)
- RFC 1662 – PPP in HDLC-Like Framing (July 1994)
- RFC 1755 – ATM Signaling Support for IP over ATM (February 1995)
- RFC 2225 – Classical IP and ARP over ATM (April 1998)
- RFC 2390 – Inverse Address Resolution Protocol (September 1998)
- RFC 2427 – Multiprotocol Interconnect over Frame Relay (September 1998)
- RFC 2464 – Transmission of IPv6 Packets over Ethernet Networks (December 1998)
- RFC 2467 – Transmission of IPv6 Packets over FDDI networks (December 1998)
- RFC 2472 – IP Version 6 over PPP (December 1998)
- RFC 2492 – IPv6 over ATM Networks (January 1999)
- RFC 2590 – Transmission of IPv6 Packets over Frame Relay Networks (May 1999)
- RFC 2615 – PPP over SONET/SDH (June 1999)
- RFC 2684 – Multiprotocol Implementation over ATM Adaptation Layer 5 (September 1999)
- RFC 3232 – Assigned Numbers: RFC 1700 is Replaced by an On-line Database (January 2002)

Archived

Archived

3

Internetworking protocols

This chapter provides an overview of the most important and common protocols associated with the TCP/IP internetwork layer. These include:

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Dynamic Host Configuration Protocol (DHCP)

These protocols perform datagram addressing, routing and delivery, dynamic address configuration, and resolve between the internetwork layer addresses and the network interface layer addresses.

© Copyright IBM Corp. 1989-2006. All rights reserved.

3.1 Internet Protocol (IP)

IP is a standard protocol with STD number 5. The standard also includes ICMP (see 3.2, “Internet Control Message Protocol (ICMP)” on page 109) and IGMP (see 3.3, “Internet Group Management Protocol (IGMP)” on page 119). IP has a status of required.

The current IP specification is in RFC 950, RFC 919, RFC 922, RFC 3260 and RFC 3168, which updates RFC 2474, and RFC 1349, which updates RFC 791. Refer to 3.8, “RFCs relevant to this chapter” on page 140 for further details regarding the RFCs.

IP is the protocol that hides the underlying physical network by creating a *virtual network* view. It is an unreliable, best-effort, and connectionless packet delivery protocol. Note that best-effort means that the packets sent by IP might be lost, arrive out of order, or even be duplicated. IP assumes higher layer protocols will address these anomalies.

One of the reasons for using a connectionless network protocol was to minimize the dependency on specific computing centers that used hierarchical connection-oriented networks. The United States Department of Defense intended to deploy a network that would still be operational if parts of the country were destroyed. This has been proven to be true for the Internet.

3.1.1 IP addressing

IP addresses are represented by a 32-bit unsigned binary value. It is usually expressed in a dotted decimal format. For example, 9.167.5.8 is a valid IP address. The numeric form is used by IP software. The mapping between the IP address and an easier-to-read symbolic name, for example, myhost.ibm.com, is done by the *Domain Name System (DNS)*, discussed in 12.1, “Domain Name System (DNS)” on page 426.

The IP address

IP addressing standards are described in RFC 1166. To identify a host on the Internet, each host is assigned an address, the *IP address*, or in some cases, the *Internet address*. When the host is attached to more than one network, it is called *multihomed* and has one IP address for each network interface. The IP address consists of a pair of numbers:

IP address = <network number><host number>

The *network number* portion of the IP address is administered by one of three Regional Internet Registries (RIR):

- American Registry for Internet Numbers (ARIN): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers for North America, South America, the Caribbean, and sub-Saharan Africa.
- Reseaux IP Europeans (RIPE): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers for Europe, Middle East, and parts of Africa.
- Asia Pacific Network Information Centre (APNIC): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers within the Asia Pacific region.

IP addresses are 32-bit numbers represented in a *dotted decimal* form (as the decimal representation of four 8-bit values concatenated with dots). For example, 128.2.7.9 is an IP address with 128.2 being the network number and 7.9 being the host number. Next, we explain the rules used to divide an IP address into its network and host parts.

The binary format of the IP address 128.2.7.9 is:

```
10000000 00000010 00000111 00001001
```

IP addresses are used by the IP protocol to uniquely identify a host on the Internet (or more generally, any internet). Strictly speaking, an IP address identifies an interface that is capable of sending and receiving IP datagrams. One system can have multiple such interfaces. However, both hosts and routers must have at least one IP address, so this simplified definition is acceptable. IP datagrams (the basic data packets exchanged between hosts) are transmitted by a physical network attached to the host. Each IP datagram contains a *source IP address* and a *destination IP address*. To send a datagram to a certain IP destination, the target IP address must be translated or mapped to a physical address. This might require transmissions in the network to obtain the destination's physical network address. (For example, on LANs, the Address Resolution Protocol, discussed in 3.4, “Address Resolution Protocol (ARP)” on page 119, is used to translate IP addresses to physical MAC addresses.)

Class-based IP addresses

The first bits of the IP address specify how the rest of the address should be separated into its network and host part. The terms *network address* and *netID* are sometimes used instead of network number, but the formal term, used in RFC 1166, is network number. Similarly, the terms *host address* and *hostID* are sometimes used instead of host number.

There are five classes of IP addresses. They are shown in Figure 3-1.

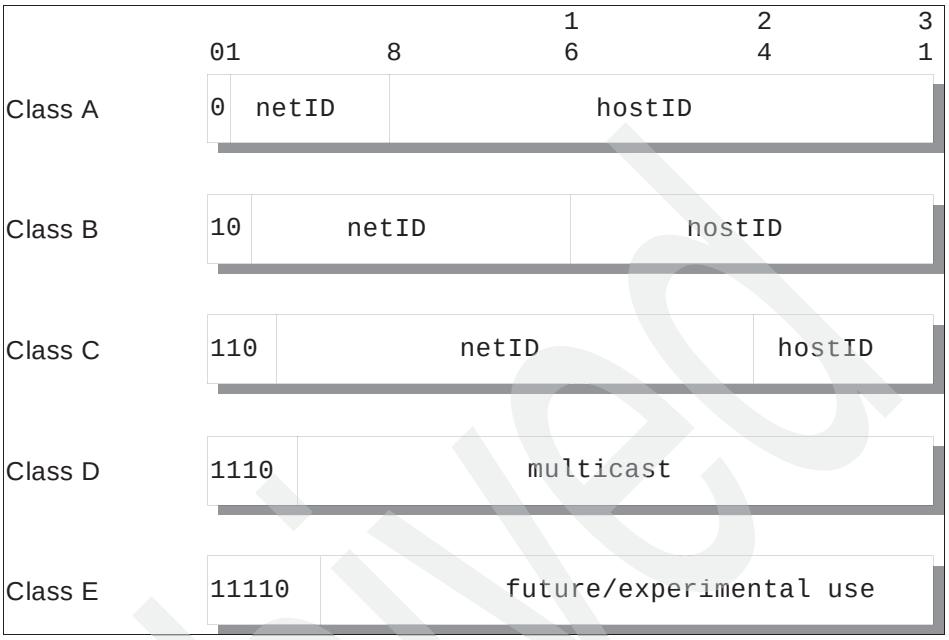


Figure 3-1 IP: Assigned classes of IP addresses

Where:

Class A addresses These addresses use 7 bits for the <network> and 24 bits for the <host> portion of the IP address. This allows for 2^7-2 (126) networks each with 2^{24}-2 (16777214) hosts—a total of more than 2 billion addresses.

Class B addresses These addresses use 14 bits for the <network> and 16 bits for the <host> portion of the IP address. This allows for 2^{14}-2 (16382) networks each with 2^{16}-2 (65534) hosts—a total of more than 1 billion addresses.

Class C addresses These addresses use 21 bits for the <network> and 8 bits for the <host> portion of the IP address. That allows for 2^{21}-2 (2097150) networks each with 2^8-2 (254) hosts—a total of more than half a billion addresses.

Class D addresses These addresses are reserved for multicasting (a sort of broadcasting, but in a limited area, and only to hosts using the same Class D address).

Class E addresses These addresses are reserved for future or experimental use.

A Class A address is suitable for networks with an extremely large number of hosts. Class C addresses are suitable for networks with a small number of hosts. This means that medium-sized networks (those with more than 254 hosts or where there is an expectation of more than 254 hosts) must use Class B addresses. However, the number of small- to medium-sized networks has been growing very rapidly. It was feared that if this growth had been allowed to continue unabated, all of the available Class B network addresses would have been used by the mid-1990s. This was termed the IP address exhaustion problem (refer to 3.1.5, “The IP address exhaustion problem” on page 86).

The division of an IP address into two parts also separates the responsibility for selecting the complete IP address. The network number portion of the address is assigned by the RIRs. The host number portion is assigned by the authority controlling the network. As shown in the next section, the host number can be further subdivided: This division is controlled by the authority that manages the network. It is not controlled by the RIRs.

Reserved IP addresses

A component of an IP address with a value *all bits 0* or *all bits 1* has a special meaning:

- All bits 0: An address with all bits zero in the host number portion is interpreted as *this* host (IP address with <host address>=0). All bits zero in the network number portion is *this* network (IP address with <network address>=0). When a host wants to communicate over a network, but does not yet know the network IP address, it can send packets with <network address>=0. Other hosts in the network interpret the address as meaning *this* network. Their replies contain the fully qualified network address, which the sender records for future use.
- All bits 1: An address with all bits one is interpreted as *all* networks or *all* hosts. For example, the following means all hosts on network 128.2 (Class B address):

  `128.2.255.255`

  This is called a directed broadcast address because it contains both a valid <network address> and a broadcast <host address>.
- Loopback: The Class A network 127.0.0.0 is defined as the loopback network. Addresses from that network are assigned to interfaces that process data within the local system. These loopback interfaces do not access a physical network.

Special use IP addresses

RFC 3330 discusses special use IP addresses. We provide a brief description of these IP addresses in Table 3-1.

Table 3-1 Special use IP addresses

Address block	Present use
0.0.0.0/8	“This” network
14.0.0.0/8	Public-data networks
24.0.0.0/8	Cable television networks
39.0.0.0/8	Reserved but subject to allocation
128.0.0.0/16	Reserved but subject to allocation
169.254.0.0/16	Link local
191.255.0.0/16	Reserved but subject to allocation
192.0.0.0/24	Reserved but subject to allocation
192.0.2.0/24	Test-Net 192.88.99.0/24 6to4 relay anycast
198.18.0.0/15	Network interconnect device benchmark testing
223.255.255.0/24	Reserved but subject to allocation
224.0.0.0/4	Multicast
240.0.0.0/4	Reserved for future use

3.1.2 IP subnets

Due to the explosive growth of the Internet, the principle of assigned IP addresses became too inflexible to allow easy changes to local network configurations. Those changes might occur when:

- A new type of physical network is installed at a location.
- Growth of the number of hosts requires splitting the local network into two or more separate networks.
- Growing distances require splitting a network into smaller networks, with gateways between them.

To avoid having to request additional IP network addresses, the concept of IP subnetting was introduced. The assignment of subnets is done locally. The entire network still appears as one IP network to the outside world.

The host number part of the IP address is subdivided into a second network number and a host number. This second network is termed a *subnetwork* or *subnet*. The main network now consists of a number of subnets. The IP address is interpreted as:

```
<network number><subnet number><host number>
```

The combination of subnet number and host number is often termed the *local address* or the *local portion* of the IP address. *Subnetting* is implemented in a way that is transparent to remote networks. A host within a network that has subnets is aware of the subnetting structure. A host in a different network is not. This remote host still regards the local part of the IP address as a host number.

The division of the local part of the IP address into a subnet number and host number is chosen by the local administrator. Any bits in the local portion can be used to form the subnet. The division is done using a 32-bit *subnet mask*. Bits with a value of zero bits in the subnet mask indicate positions ascribed to the host number. Bits with a value of one indicate positions ascribed to the subnet number. The bit positions in the subnet mask belonging to the original network number are set to ones but are not used (in some platform configurations, this value was specified with zeros instead of ones, but either way it is not used). Like IP addresses, subnet masks are usually written in dotted decimal form.

The special treatment of all bits zero and all bits one applies to each of the three parts of a subnetted IP address just as it does to both parts of an IP address that has not been subnetted (see “Reserved IP addresses” on page 71). For example, subnetting a Class B network can use one of the following schemes:

- The first octet is the subnet number; the second octet is the host number. This gives 2^8-2 (254) possible subnets, each having up to 2^8-2 (254) hosts. Recall that we subtract two from the possibilities to account for the all ones and all zeros cases. The subnet mask is 255.255.255.0.
- The first 12 bits are used for the subnet number and the last four for the host number. This gives 2^{12}-2 (4094) possible subnets but only 2^4-2 (14) hosts per subnet. The subnet mask is 255.255.255.240.

In this example, there are several other possibilities for assigning the subnet and host portions of the address. The number of subnets and hosts and any future requirements need to be considered before defining this structure. In the last example, the subnetted Class B network has 16 bits to be divided between the subnet number and the host number fields. The network administrator defines either a larger number of subnets each with a small number of hosts, or a smaller number of subnets each with many hosts.

When assigning the subnet part of the local address, the objective is to assign a *number* of bits to the subnet number and the remainder to the local address.

Therefore, it is normal to use a contiguous block of bits at the beginning of the local address part for the subnet number. This makes the addresses more readable. (This is particularly true when the subnet occupies 8 or 16 bits.) With this approach, either of the previous subnet masks are “acceptable” masks. Masks such as 255.255.252.252 and 255.255.255.15 are “unacceptable.” In fact, most TCP/IP implementations do not support non-contiguous subnet masks. Their use is universally discouraged.

Types of subnetting

There are two types of subnetting: static and variable length. Variable length subnetting is more flexible than static. Native IP routing and RIP Version 1 support only static subnetting. However, RIP Version 2 supports variable length subnetting (refer to Chapter 5, “Routing protocols” on page 171).

Static subnetting

Static subnetting implies that all subnets obtained from the same network use the same subnet mask. Although this is simple to implement and easy to maintain, it might waste address space in small networks. Consider a network of four hosts using a subnet mask of 255.255.255.0. This allocation wastes 250 IP addresses. All hosts and routers are required to support static subnetting.

Variable length subnetting

When variable length subnetting or variable length subnet masks (VLSM) are used, allocated subnets within the same network can use different subnet masks. A small subnet with only a few hosts can use a mask that accommodates this need. A subnet with many hosts requires a different subnet mask. The ability to assign subnet masks according to the needs of the individual subnets helps conserve network addresses. Variable length subnetting divides the network so that each subnet contains sufficient addresses to support the required number of hosts.

An existing subnet can be split into two parts by adding another bit to the subnet portion of the subnet mask. Other subnets in the network are unaffected by the change.

Mixing static and variable length subnetting

Not every IP device includes support for variable length subnetting. Initially, it appears that the presence of a host that only supports static subnetting prevents the use of variable length subnetting. This is not the case. Routers interconnecting the subnets are used to hide the different masks from hosts. Hosts continue to use basic IP routing. This offloads subnetting complexities to dedicated routers.

Static subnetting example

Consider the Class A network shown in Figure 3-2.

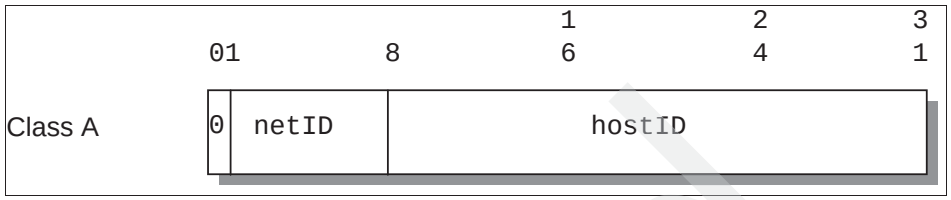


Figure 3-2 IP: Class A address without subnets

Use the IP address shown in Figure 3-3.

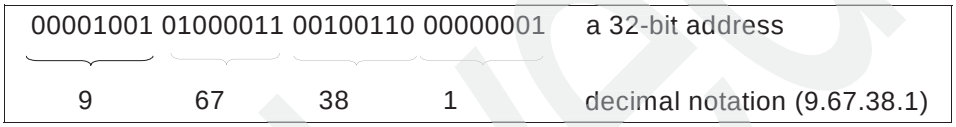


Figure 3-3 IP address

The IP address is 9.67.38.1 (Class A) with 9 as the <network address> and 67.38.1 as the <host address>.

The network administrator might want to choose the bits from 8 to 25 to indicate the subnet address. In that case, the bits from 26 to 31 indicate the host addresses. Figure 3-4 shows the subnetted address derived from the original Class A address.

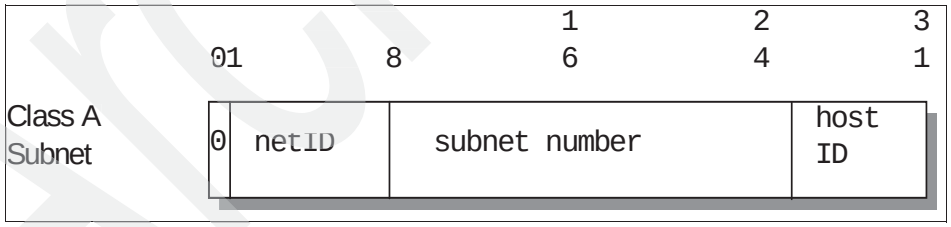


Figure 3-4 IP: Class A address with subnet mask and subnet address

A bit mask, known as the subnet mask, is used to identify which bits of the original host address field indicate the subnet number. In the previous example, the subnet mask is 255.255.255.192 (or 11111111 11111111 11111111 11000000 in bit notation). Note that, by convention, the <network address> is included in the mask as well.

Because of the all bits 0 and all bits 1 restrictions, this defines 2^{18}-2 (from 1 to 262143) valid subnets. This split provides 262142 subnets each with a maximum of 2^6-2 (62) hosts.

The value applied to the subnet number takes the value of the full octet with non-significant bits set to zero. For example, the hexadecimal value 01 in this subnet mask assumes an 8-bit value 01000000. This provides a subnet value of 64.

Applying the 255.255.255.192 to the sample Class A address of 9.67.38.1 provides the following information:

```
00001001 01000011 00100110 00000001 = 9.67.38.1 (Class A address)
   11111111 11111111 11111111 11------   255.255.255.192 (subnet mask)
  ===================================  logical_AND
   00001001 01000011 00100110 00------ = 9.67.38.0 (subnet base address)
```

This leaves a host address of:

```
   -------- -------- -------- --000001 = 1        (host address)
```

IP will recognize all host addresses as being on the local network for which the logical_AND operation described earlier produces the same result. This is important for routing IP datagrams in subnet environments (refer to 3.1.3, “IP routing” on page 77).

The subnet number is:

```
   -------- 01000011 00100110 00------ = 68760    (subnet number)
```

This subnet number is a relative number. That is, it is the 68760th subnet of network 9 with the given subnet mask. This number bears no resemblance to the actual IP address that this host has been assigned (9.67.38.1). It has no meaning in terms of IP routing.

The division of the original <host address> into <subnet><host> is chosen by the network administrator. The values of all zeroes and all ones in the <subnet> field are reserved.

Variable length subnetting example

Consider a corporation that has been assigned the Class C network 165.214.32.0. The corporation has the requirement to split this address range into five separate networks each with the following number of hosts:

- Subnet 1: 50 hosts
- Subnet 2: 50 hosts
- Subnet 3: 50 hosts

- Subnet 4: 30 hosts
- Subnet 5: 30 hosts

This cannot be achieved with static subnetting. For this example, static subnetting divides the network into four subnets each with 64 hosts or eight subnets each with 32 hosts. This subnet allocation does not meet the stated requirements.

To divide the network into five subnets, multiple masks need to be defined. Using a mask of 255.255.255.192, the network can be divided into four subnets each with 64 hosts. The fourth subnet can be further divided into two subnets each with 32 hosts by using a mask of 255.255.255.224. There will be three subnets each with 64 hosts and two subnets each with 32 hosts. This satisfies the stated requirements and eliminates the possibility of a high number of wasted host addresses.

Determining the subnet mask

Usually, hosts will store the subnet mask in a configuration file. However, sometimes this cannot be done, for example, as in the case of a diskless workstation. The ICMP protocol includes two messages: address mask request and address mask reply. These allow hosts to obtain the correct subnet mask from a server (refer to “Address Mask Request (17) and Address Mask Reply (18)” on page 116).

Addressing routers and multihomed hosts

Whenever a host has a physical connection to multiple networks or subnets, it is described as being *multihomed*. By default, all routers are multihomed because their purpose is to join networks or subnets. A multihomed host has different IP addresses associated with each network adapter. Each adapter connects to a different subnet or network.

3.1.3 IP routing

An important function of the IP layer is *IP routing*. This provides the basic mechanism for routers to interconnect different physical networks. A device can simultaneously function as both a normal host and a router.

A router of this type is referred to as a router with partial routing information. The router only has information about four kinds of destinations:

- Hosts that are directly attached to one of the physical networks to which the router is attached.
- Hosts or networks for which the router has been given explicit definitions.

- Hosts or networks for which the router has received an ICMP redirect message.
- A default for all other destinations.

Additional protocols are needed to implement a full-function router. These types of routers are essential in most networks, because they can exchange information with other routers in the environment. We review the protocols used by these routers in Chapter 5, “Routing protocols” on page 171.

There are two types of IP routing: direct and indirect.

Direct routing

If the destination host is attached to the same physical network as the source host, IP datagrams can be directly exchanged. This is done by encapsulating the IP datagram in the physical network frame. This is called direct delivery and is referred to as direct routing.

Indirect routing

Indirect routing occurs when the destination host is not connected to a network directly attached to the source host. The only way to reach the destination is through one or more IP gateways. (Note that in TCP/IP terminology, the terms gateway and router are used interchangeably. This describes a system that performs the duties of a router.) The address of the first gateway (the first hop) is called an indirect route in the IP routing algorithm. The address of the first gateway is the only information needed by the source host to send a packet to the destination host.

In some cases, there may be multiple subnets defined on the same physical network. If the source and destination hosts connect to the same physical network but are defined in different subnets, indirect routing is used to communicate between the pair of devices. A router is needed to forward traffic between subnets.

Archived

Figure 3-5 shows an example of direct and indirect routes. Here, host C has a direct route to hosts B and D, and an indirect route to host A via gateway B.

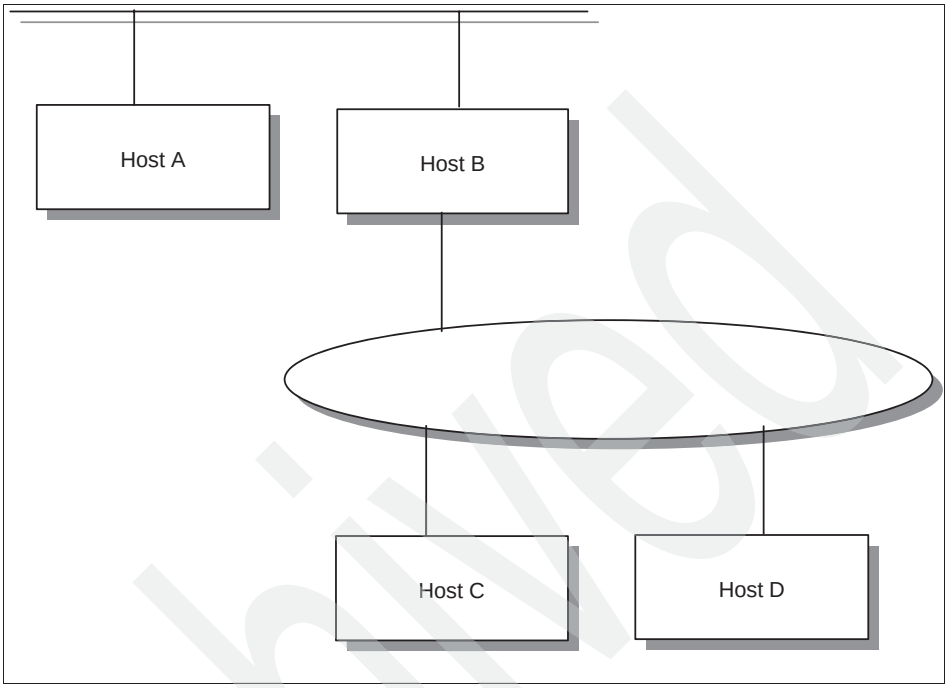


Figure 3-5 IP: Direct and indirect routes

IP routing table

The determination of direct routes is derived from the list of local interfaces. It is automatically composed by the IP routing process at initialization. In addition, a list of networks and associated gateways (indirect routes) can be configured. This list is used to facilitate IP routing. Each host keeps the set of mappings between the following:

- Destination IP network addresses
- Routes to next gateways

This information is stored in a table called the IP routing table. Three types of mappings are in this table:

- The direct routes describing locally attached networks
- The indirect routes describing networks reachable through one or more gateways

Archived

- The default route that contains the (direct or indirect) route used when the destination IP network is not found in the mappings of the previous types of type 1 and 2

Figure 3-6 presents a sample network.

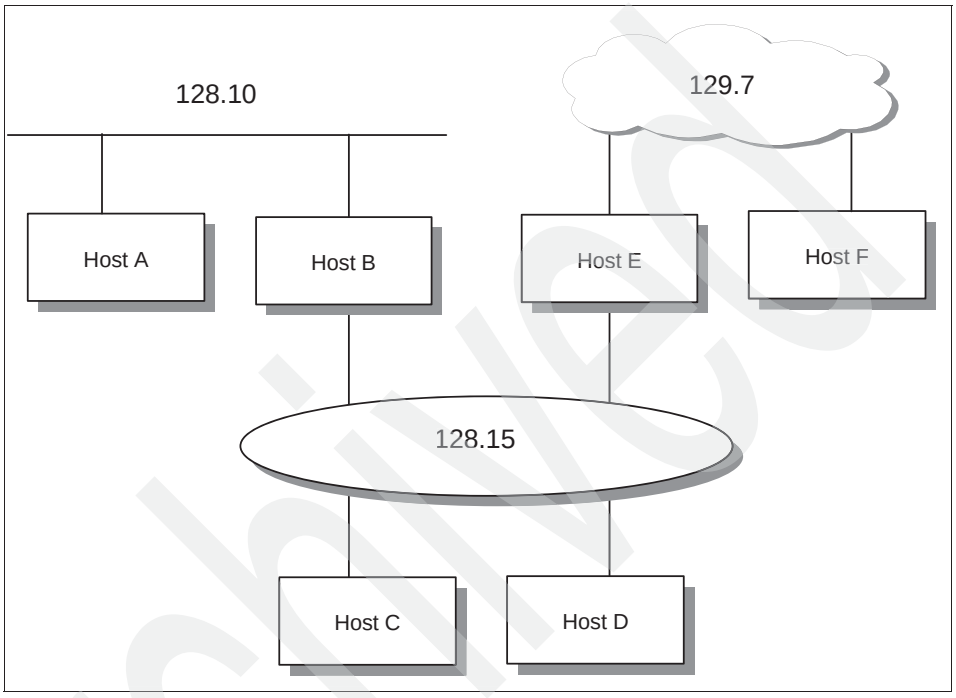


Figure 3-6 IP: Routing table scenario

The routing table of host D might contain the following (symbolic) entries (Table 3-2).

Table 3-2 Host D sample entries

Destination	Router	Interface
129.7.0.0	E	lan0
128.15.0.0	D	lan0
128.10.0.0	B	lan0
default	B	lan0
127.0.0.1	loopback	loo

Archived

Because D is directly attached to network 128.15.0.0, it maintains a direct route for this network. To reach networks 129.7.0.0 and 128.10.0.0, however, it must have an indirect route through E and B, respectively, because these networks are not directly attached to it.

The routing table of host F might contain the following (symbolic) entries (Table 3-3).

Table 3-3 Host F sample entries

Destination	Router	Interface
129.7.0.0	F	wan0
default	E	wan0
127.0.0.1	loopback	lo

Because every host not on the 129.7.0.0 network must be reached through host E, host F simply maintains a default route through E.

IP routing algorithm

IP uses a unique algorithm to route datagrams, as illustrated in Figure 3-7.

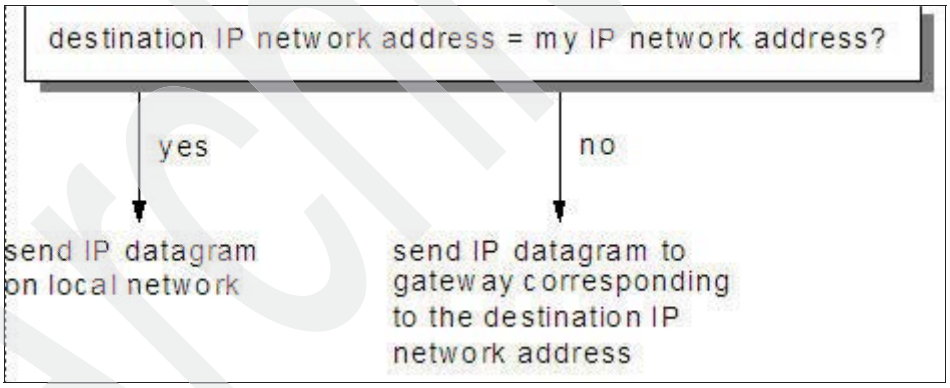


Figure 3-7 IP: Routing without subnets

Archived

To differentiate between subnets, the IP routing algorithm is updated, as shown in Figure 3-8.

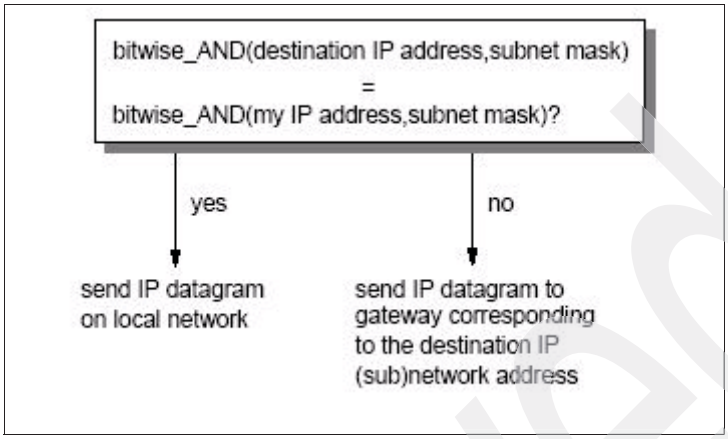


Figure 3-8 IP: Routing with subnets

Some implications of this change include:

- This algorithm represents a change to the general IP algorithm. Therefore, to be able to operate this way, the particular gateway must contain the new algorithm. Some implementations might still use the general algorithm, and will not function within a subnetted network, although they can still communicate with hosts in other networks that are subnetted.
- As IP routing is used in all of the hosts (and not just the routers), all of the hosts in the subnet must have:
  - An IP routing algorithm that supports subnetting
  - The same subnet mask (unless subnets are formed within the subnet)
- If the IP implementation on any of the hosts does not support subnetting, that host will be able to communicate with any host in its own subnet but not with any machine on another subnet within the same network. This is because the host sees only one IP network and its routing cannot differentiate between an IP datagram directed to a host on the local subnet and a datagram that should be sent through a router to a different subnet.

In case one or more hosts do not support subnetting, an alternative way to achieve the same goal exists in the form of *proxy-ARP*. This does not require any changes to the IP routing algorithm for single-homed hosts. It does require changes on routers between subnets in the network (refer to 3.4.4, “Proxy-ARP or transparent subnetting” on page 123).

Figure 3-9 illustrates the entire IP routing algorithm.

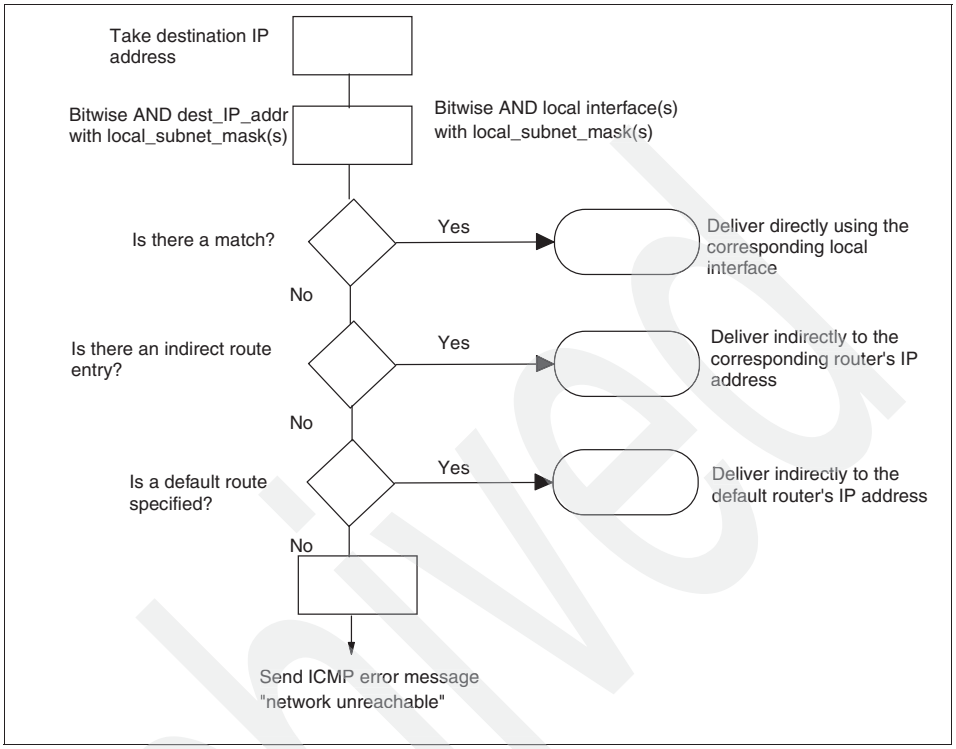


Figure 3-9 IP: Routing algorithm (with subnets)

Archived

3.1.4 Methods of delivery: Unicast, broadcast, multicast, and anycast

The majority of IP addresses refer to a single recipient, this is called a *unicast* address. Unicast connections specify a one-to-one relationship between a single source and a single destination. Additionally, there are three special types of IP addresses used for addressing multiple recipients: broadcast addresses, multicast addresses, and anycast addresses. Figure 3-10 shows their operation.

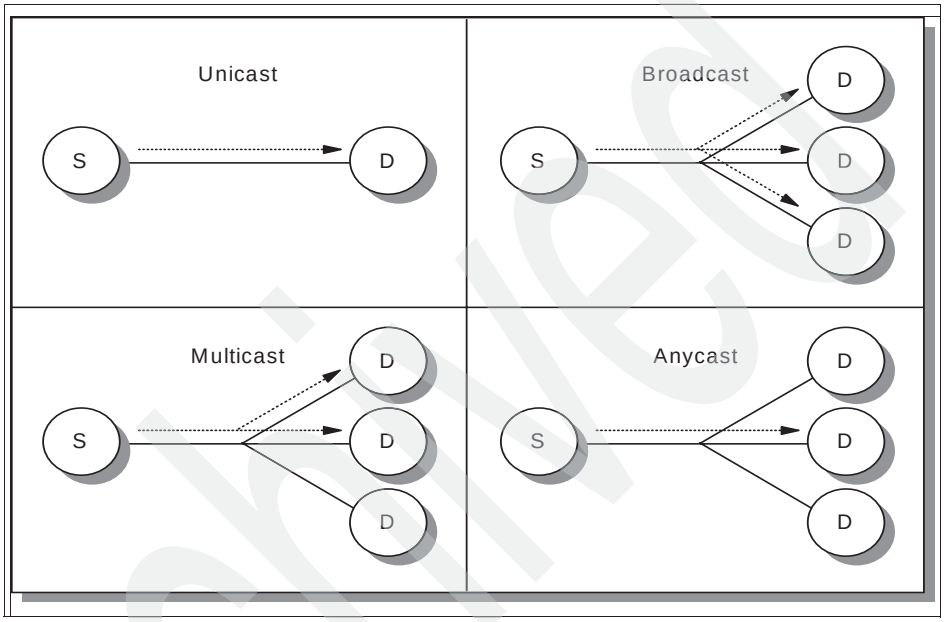


Figure 3-10 IP: Packet delivery modes

A *connectionless* protocol can send unicast, broadcast, multicast, or anycast messages. A *connection-oriented* protocol can only use unicast addresses (a connection must exist between a specific pair of hosts).

Broadcasting

Broadcast addresses are never valid as a source address. They must specify the destination address. The different types of broadcast addresses include:

- Limited broadcast address: This uses the address 255.255.255.255 (all bits 1 in all parts of the IP address). It refers to all hosts on the local subnet. This is recognized by every host. The hosts do not need any IP configuration information. Routers do not forward this packet.

  One exception to this rule is called *BOOTP forwarding*. The BOOTP protocol uses the limited broadcast address to allow a diskless workstation to contact a boot server. BOOTP forwarding is a configuration option available on some

routers. Without this facility, a separate BOOTP server is required on each subnet (refer to 3.6, “Bootstrap Protocol (BOOTP)” on page 125).

- Network-directed broadcast address: This is used in an unsubnetted environment. The network number is a valid network number and the host number is all ones (for example, 128.2.255.255). This address refers to all hosts on the specified network. Routers should forward these broadcast messages. This is used in ARP requests (refer to 3.4, “Address Resolution Protocol (ARP)” on page 119) on unsubnetted networks.
- Subnet-directed broadcast address: If the network number is a valid network number, the subnet number is a valid subnet number, and the host number is all ones, the address refers to all hosts on the specified subnet. Because the sender's subnet and the target subnet might have a different subnet mask, the sender must somehow determine the subnet mask in use at the target. The broadcast is performed by the router that receives the datagram into the subnet.
- All-subnets-directed broadcast address: If the network number is a valid network number, the network is subnetted, and the local part is all ones (for example, 128.2.255.255), the address refers to all hosts on all subnets in the specified network. In principle, routers can propagate broadcasts for all subnets but are not required to do so. In practice, they do not. There are very few circumstances where such a broadcast is desirable. If misconfigured, it can lead to problems. Consider the misconfigured host 9.180.214.114 in a subnetted Class A network. If the device was configured with the address 9.255.255.255 as a local broadcast address instead of 9.180.214.255, all of the routers in the network will forward the request to all clients.

  If routers do respect all-subnets-directed broadcast address, they use an algorithm called *reverse path forwarding* to prevent the broadcast messages from multiplying out of control. See RFC 922 for more details about this algorithm.

Multicasting

If an IP datagram is broadcast to a subnet, it is received by every host on the subnet. Each host processes the packet to determine if the target protocol is active. If it is not active, the IP datagram is discarded. Multicasting avoids this by selecting destination groups.

Each group is represented by a Class D IP address. For each multicast address, a set of zero or more hosts are listening for packets addressed to the address. This set of hosts is called the *host group*. Packets sent to a multicast address are forwarded only to the members of the corresponding host group. Multicast enables one-to-many connections (refer to Chapter 6, “IP multicast” on page 237).

Anycasting

Sometimes, the same IP services are provided by different hosts. For example, a user wants to download a file using FTP and the file is available on multiple FTP servers. Hosts that implement the same service provide an anycast address to other hosts that require the service. Connections are made to the first host in the anycast address group to respond. This process is used to guarantee the service is provided by the host with the best connection to the receiver.

The anycast service is included in IPV6 (refer to 9.2.2, “IPv6 addressing” on page 339).

3.1.5 The IP address exhaustion problem

The number of networks on the Internet has been approximately doubling annually for a number of years. However, the usage of the Class A, B, and C networks differs greatly. Nearly all of the new networks assigned in the late 1980s were Class B, and in 1990 it became apparent that if this trend continued, the last Class B network number would be assigned during 1994. However, Class C networks were hardly being used.

The reason for this trend was that most potential users found a Class B network to be large enough for their anticipated needs, because it accommodates up to 65534 hosts, while a Class C network, with a maximum of 254 hosts, severely restricts the potential growth of even a small initial network. Furthermore, most of the Class B networks being assigned were small ones. There are relatively few networks that would need as many as 65,534 host addresses, but very few for which 254 hosts would be an adequate limit. In summary, although the Class A, Class B, and Class C divisions of the IP address are logical and easy-to-use (because they occur on byte boundaries), with hindsight, they are not the most practical because Class C networks are too small to be useful for most organizations, while Class B networks are too large to be densely populated by any but the largest organizations.

In May 1996, all Class A addresses were either allocated or assigned, as well as 61.95 percent of Class B and 36.44 percent of Class C IP network addresses. The terms assigned and allocated in this context have the following meanings:

- Assigned: The number of network numbers in use. The Class C figures are somewhat inaccurate, because the figures do not include many Class C networks in Europe, which were allocated to RIPE and subsequently assigned but which are still recorded as allocated.
- Allocated: This includes all of the assigned networks and additionally, those networks that have either been reserved by IANA (for example, the 63 Class A networks are all reserved by IANA) or have been allocated to regional registries by IANA and will subsequently be assigned by those registries.

Another way to look at these numbers is to examine the proportion of the address space that has been used. For example, the Class A address space is as big as the rest combined, and a single Class A network can theoretically have as many hosts as 66,000 Class C networks.

Since 1990, the number of assigned Class B networks has been increasing at a much lower rate than the total number of assigned networks and the anticipated exhaustion of the Class B network numbers has not yet occurred. The reason for this is that the policies on network number allocation were changed in late 1990 to preserve the existing address space, in particular to avert the exhaustion of the Class B address space. The new policies can be summarized as follows:

- The upper half of the Class A address space (network numbers 64 to 127) is reserved indefinitely to allow for the possibility of using it for transition to a new numbering scheme.
- Class B networks are only assigned to organizations that can clearly demonstrate a need for them. The same is, of course, true for Class A networks. The requirements for Class B networks are that the requesting organization:
  - Has a subnetting plan that documents more than 32 subnets within its organizational network
  - Has more than 4096 hosts

  Any requirements for a Class A network are handled on an individual case basis.
- Organizations that do not fulfill the requirements for a Class B network are assigned a consecutively numbered block of Class C network numbers.
- The lower half of the Class C address space (network numbers 192.0.0 through 207.255.255) is divided into eight blocks, which are allocated to regional authorities as follows:

  192.0.0 - 193.255.255 Multi-regional

  194.0.0 - 195.255.255 Europe

  196.0.0 - 197.255.255 Others

  198.0.0 - 199.255.255 North America

  200.0.0 - 201.255.255 Central and South America

  202.0.0 - 203.255.255 Pacific Rim

  204.0.0 - 205.255.255 Others

  206.0.0 - 207.255.255 Others

  208.0.0 - 209.255.255 ARIN[1]

  210.0.0 - 211.255.255 APNIC

212.0.0 - 213.255.255 RIPE NCC

214.0.0 - 215.255.255 US Department of Defense

216.0.0 - 216.255.255 ARIN

217.0.0 - 217.255.255 RIPE NCC

218.0.0 - 218.255.255 APNIC

219.0.0 - 222.255.255 APNIC

The ranges defined as Others are to be where flexibility outside the constraints of regional boundaries is required. The range defined as multi-regional includes the Class C networks that were assigned before this new scheme was adopted. The 192 networks were assigned by the InterNIC and the 193 networks were previously allocated to RIPE in Europe.

- Where an organization has a range of Class C network numbers, the range provided is assigned as a *bit-wise contiguous* range of network numbers, and the number of networks in the range is a power of 2. That is, all IP addresses in the range have a common prefix, and every address with that prefix is within the range. For example, a European organization requiring 1500 IP addresses would be assigned eight Class C network numbers (2048 IP addresses) from the number space reserved for European networks (194.0.0 through 195.255.255) and the first of these network numbers would be divisible by eight. A range of addresses satisfying these rules is 194.32.136 through 194.32.143, in which case the range consists of all of the IP addresses with the 21-bit prefix 194.32.136, or B '11000010001000010001'.

  The maximum number of network numbers assigned contiguously is 64, corresponding to a prefix of 18 bits. An organization requiring more than 4096 addresses but less than 16,384 addresses can request either a Class B or a range of Class C addresses. In general, the number of Class C networks assigned is the minimum required to provide the necessary number of IP addresses for the organization on the basis of a two-year outlook. However, in some cases, an organization can request multiple networks to be treated separately. For example, an organization with 600 hosts is normally assigned four Class C networks. However, if those hosts were distributed across 10 LANs with between 50 and 70 hosts per LAN, such an allocation can cause serious problems, because the organization would have to find 10 subnets within a 10-bit local address range. This means at least some of the LANs have a subnet mask of 255.255.255.192, which allows only 62 hosts per LAN. The intent of the rules is not to force the organization into complex subnetting of small networks, and the organization should request 10 different Class C numbers, one for each LAN.

[1] Information for this and the following numbers in this list are from: http://www.iana.org/assignments/ipv4-address-space

The current rules are in RFC 2050, which updates RFC 1466. The reasons for the rules for the allocation of Class C network numbers will become apparent in the following sections. The use of Class C network numbers in this way has averted the exhaustion of the Class B address space, but it is not a permanent solution to the overall address space constraints that are fundamental to IP. We discuss a long-term solution in Chapter 9, “IP version 6” on page 327.

3.1.6 Intranets: Private IP addresses

Another approach to conserve the IP address space is described in RFC 1918. This RFC relaxes the rule that IP addresses must be globally unique. It reserves part of the global address space for use in networks that do not require connectivity to the Internet. Typically these networks are administered by a single organization. Three ranges of addresses have been reserved for this purpose:

- 10.0.0.0: A single Class A network
- 172.16.0.0 through 172.31.0.0: 16 contiguous Class B networks
- 192.168.0.0 through 192.168.255.0: 256 contiguous Class C networks

Any organization can use any address in these ranges. However, because these addresses are not globally unique, they are not defined to any external routers. Routers in networks not using private addresses, particularly those operated by Internet service providers, are expected to quietly discard all routing information regarding these addresses. Routers in an organization using private addresses are expected to limit all references to private addresses to internal links. They should neither externally advertise routes to private addresses nor forward IP datagrams containing private addresses to external routers.

Hosts having only a private IP address do not have direct IP layer connectivity to the Internet. All connectivity to external Internet hosts must be provided with application gateways (refer to “Application-level gateway (proxy)” on page 798), SOCKS (refer to 22.5, “SOCKS” on page 846), or Network Address Translation (NAT), which is discussed in the next section.

3.1.7 Network Address Translation (NAT)

This section explains Traditional Network Address Translation (NAT), Basic NAT, and Network Address Port Translation (NAPT). NAT is also known as IP masquerading. It provides a mapping between internal IP addresses and officially assigned external addresses.

Originally, NAT was suggested as a short-term solution to the IP address exhaustion problem. Also, many organizations have, in the past, used locally assigned IP addresses, not expecting to require Internet connectivity.

There are two variations of traditional NAT, Basic NAT and NAPT. Traditional NAT is defined in RFC 3022 and discussed in RFC 2663. The following sections provide a brief discussion of Traditional NAT, Basic NAT, and NAPT based on RFC 3022.

Traditional NAT

The idea of Traditional NAT (hereafter referred to as NAT) is based on the fact that only a small number of the hosts in a private network are communicating outside of that network. If each host is assigned an IP address from the official IP address pool only when they need to communicate, only a small number of official addresses are required.

NAT might be a solution for networks that have private address ranges or unofficial addresses and want to communicate with hosts on the Internet. When a proxy server, SOCKS server, or firewall are not available, or do not meet specific requirements, NAT might be used to manage the traffic between the internal and external network without advertising the internal host addresses.

Basic NAT

Consider an internal network that is based on the private IP address space, and the users want to use an application protocol for which there is no application gateway; the only option is to establish IP-level connectivity between hosts in the internal network and hosts on the Internet. Because the routers in the Internet would not know how to route IP packets back to a private IP address, there is no point in sending IP packets with private IP addresses as source IP addresses through a router into the Internet.

Archived

As shown in Figure 3-11, Basic NAT takes the IP address of an outgoing packet and dynamically translates it to an officially assigned global address. For incoming packets, it translates the assigned address to an internal address.

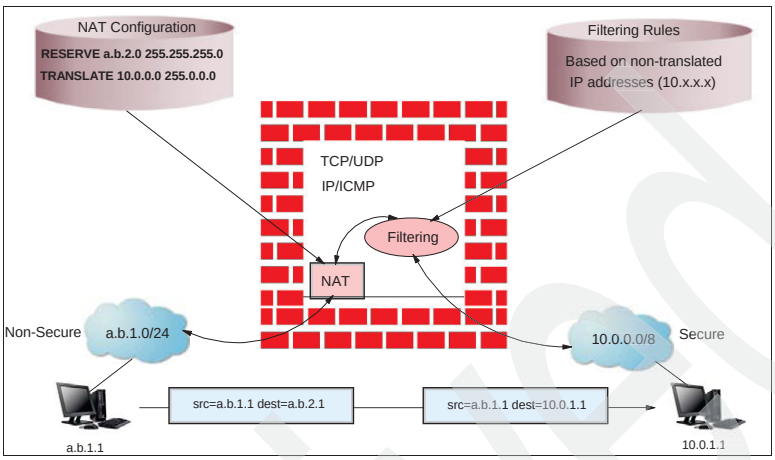


Figure 3-11 Basic Network Address Translation (NAT)

From the point of two hosts that exchange IP packets with each other, one in the internal network and one in the external network, the NAT itself is transparent (see Figure 3-12).

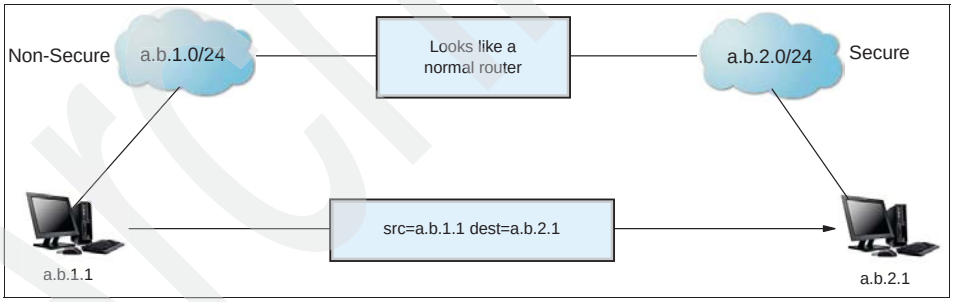


Figure 3-12 NAT seen from the external network

Basic NAT translation mechanism

For each outgoing IP packet, the source address is checked by the NAT configuration rules. If a rule matches the source address, the address is translated to a global address from the address pool. The predefined address pool contains the addresses that NAT can use for translation. For each incoming

packet, the destination address is checked if it is used by NAT. When this is true, the address is translated to the original internal address. Figure 3-13 shows the Basic NAT configuration.

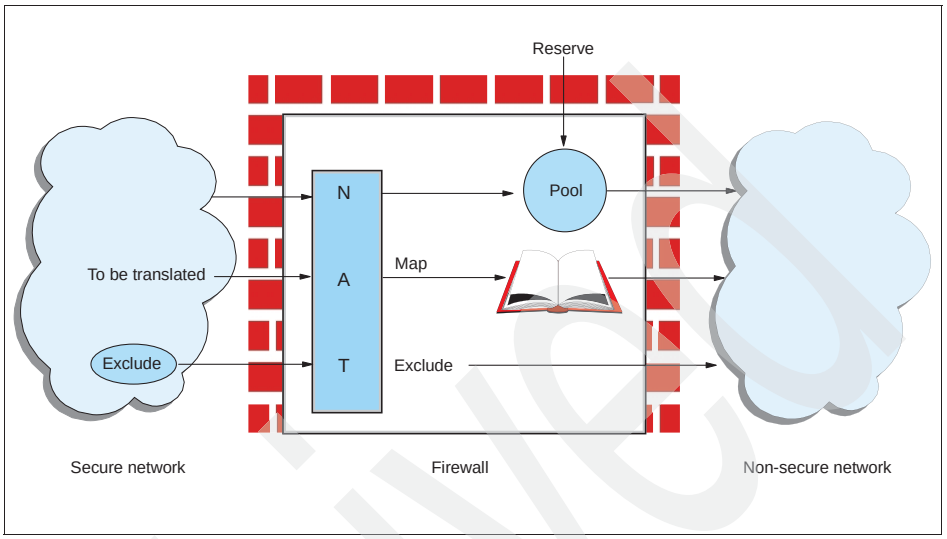


Figure 3-13 Basic NAT configuration

When Basic NAT translates an address for an IP packet, the checksum is also adjusted. For FTP packets, the task is even more difficult, because the packets can contain addresses in the data of the packet. For example, the FTP PORT command contains an IP address in ASCII. These addresses should also be translated correctly; checksum updates and TCP sequence and acknowledgement updates should also be made accordingly.

In order to make the routing tables work, the IP network design needs to choose addresses as though connecting two or more IP networks or subnets through a router. The NAT IP addresses need to come from separate networks or subnets, and the addresses need to be unambiguous with respect to other networks or subnets in the non-secure network. If the external network is the Internet, the NAT addresses need to come from a public network or subnet; in other words, the NAT addresses need to be assigned by IANA.

The assigned addresses need to be reserved in a pool in order to use them when needed. If connections are established from the internal network, NAT can just pick the next free public address in the NAT pool and assign that to the requesting internal host. The NAT service keeps track of which internal IP addresses are mapped to which external IP addresses at any given point in time, so it will be able to map a response it receives from the external network into the corresponding secure IP address.

When the NAT service assigns IP addresses on a demand basis, it needs to know when to return the external IP address to the pool of available IP addresses. There is no connection setup or tear-down at the IP level, so there is nothing in the IP protocol itself that the NAT service can use to determine when an association between a internal IP address and a NAT external IP address is no longer needed. Because TCP is a connection-oriented protocol, it is possible to obtain the connection status information from TCP header (whether connection is ended or not), while UDP does not include such information. Therefore, configure a timeout value that instructs NAT how long to keep an association in an idle state before returning the external IP address to the free NAT pool. Generally, the default value for this parameter is 15 minutes.

Network administrators also need to instruct NAT whether all the internal hosts are allowed to use NAT or not. This can be done by using corresponding configuration commands. If hosts in the external network need to initiate connections to hosts in the internal network, NAT needs to be configured in advance as to which external NAT address matches which internal IP address. Thus, a static mapping should be defined to allow connections from outside networks to a specific host in the internal network. Note that the external NAT addresses as statically mapped to internal IP addresses should not overlap with the addresses specified as belonging to the pool of external addresses that the NAT service can use on a demand basis.

The external name server can, for example, have an entry for a mail gateway that runs on a computer in the internal network. The external name server resolves the public host name of the internal mail gateway to the statically mapped IP address (the external address), and the remote mail server sends a connection request to this IP address. When that request comes to the NAT service on the external interface, the NAT service looks into its mapping rules to see if it has a static mapping between the specified external public IP address and a internal IP address. If so, it translates the IP address and forwards the IP packet into the internal network to the mail gateway.

Network Address Port Translation (NAPT)

The difference between Basic NAT and NAPT is that Basic NAT is limited to only translating IP addresses, while NAPT is extended to include IP address and transport identifier (such as TCP/UDP port or ICMP query ID).

As shown in Figure 3-14, Network Address Port Translation is able to translate many network addresses and their transport identifiers into a single network address with many transport identifiers, or more specifically, ports.

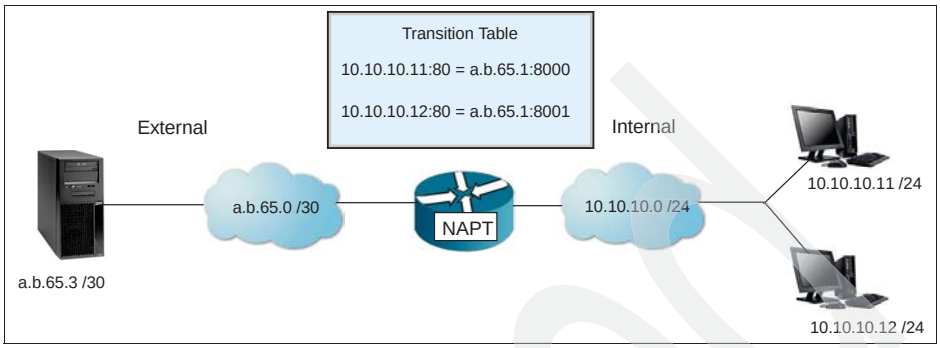


Figure 3-14 Network Address Port Translation

NAPT maps private addresses to a single globally unique address. Therefore, the binding is from the private address and private port to the assigned address and assigned port. NAPT permits multiple nodes in a local network to simultaneously access remote networks using the single IP address assigned to their router.

In NAPT, modifications to the IP header are similar to that of Basic NAT. However for TCP/UDP sessions, modifications must be extended to include translation of the source port for outbound packets and destination port for inbound packets in the TCP/UDP header. In addition to TCP/UDP sessions, ICMP messages, with the exception of the REDIRECT message type, can also be monitored by the NAPT service running on the router. ICMP query type packets are translated similar to that of TCP/UDP packets in that the identifier field in ICMP message header will be uniquely mapped to a query identifier of the registered IP address.

NAT limitations

The NAT limitations are mentioned in RFC 3022 and RFC2663. We discuss some of the limitations here.

NAT works fine for IP addresses in the IP header. Some application protocols exchange IP address information in the application data part of an IP packet, and NAT will generally not be able to handle translation of IP addresses in the application protocol. Currently, most of the implementations handle the FTP protocol. It should be noted that implementation of NAT for specific applications that have IP information in the application data is more sophisticated than the standard NAT implementations.

NAT is compute intensive even with the assistance of a sophisticated checksum adjustment algorithm, because each data packet is subject to NAT lookup and modifications.

It is mandatory that all requests and responses pertaining to a session be routed through the same router that is running the NAT service.

Translation of outbound TCP/UDP fragments (that is, those originating from private hosts) in a NAPT setup will not work (refer to “Fragmentation” on page 104). This is because only the first fragment contains the TCP/UDP header that is necessary to associate the packet to a session for translation purposes. Subsequent fragments do not contain TCP/UDP port information, but simply carry the same fragmentation identifier specified in the first fragment. When the target host receives the two unrelated datagrams, carrying the same fragmentation ID and from the same assigned host address, it is unable to determine to which of the two sessions the datagrams belong. Consequently, both sessions will be corrupted.

NAT changes some of the address information in an IP packet. This becomes an issue when IPSec is used. Refer to 22.4, “IP Security Architecture (IPSec)” on page 809 and 22.10, “Virtual private networks (VPNs) overview” on page 861. When end-to-end IPSec authentication is used, a packet whose address has been changed will always fail its integrity check under the Authentication Header protocol, because any change to any bit in the datagram will invalidate the integrity check value that was generated by the source. Because IPSec protocols offer some solutions to the addressing issues that were previously handled by NAT, there is no need for NAT when all hosts that compose a given virtual private network use globally unique (public) IP addresses. Address hiding can be achieved by the IPSec tunnel mode. If a company uses private addresses within its intranet, the IPSec tunnel mode can keep them from ever appearing in cleartext from in the public Internet, which eliminates the need for NAT.

3.1.8 Classless Inter-Domain Routing (CIDR)

Standard IP routing understands only Class A, B, and C network addresses. Within each of these networks, subnetting can be used to provide better granularity. However, there is no way to specify that multiple Class C networks are related. The result of this is termed the *routing table explosion* problem: A Class B network of 3000 hosts requires one routing table entry at each backbone router. The same environment, if addressed as a range of Class C networks, requires 16 entries.

The solution to this problem is called Classless Inter-Domain Routing (CIDR). CIDR is described in RFCs 1518 to 1520. CIDR does not route according to the

class of the network number (thus the term classless). It is based solely on the high order bits of the IP address. These bits are called the IP prefix.

Each CIDR routing table entry contains a 32-bit IP address and a 32-bit network mask, which together give the length and value of the IP prefix. This is represented as the tuple <IP_address network_mask>. For example, to address a block of eight Class C addresses with one single routing table entry, the following representation suffices: <192.32.136.0 255.255.248.0>. This refers, from a backbone point of view, to the Class C network range from 192.32.136.0 to 192.32.143.0 as one single network. This is illustrated in Figure 3-15.

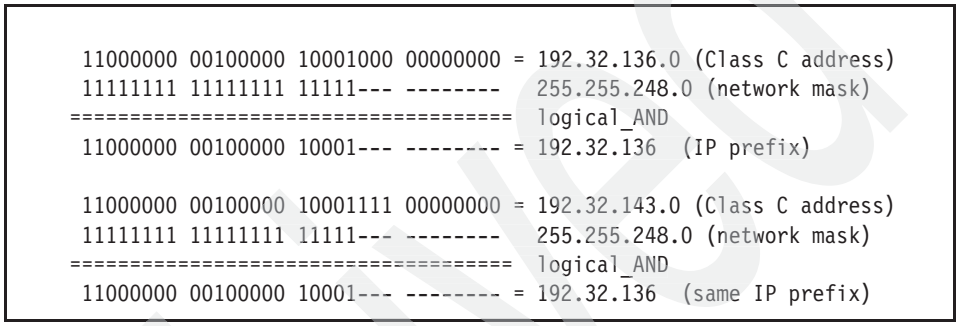


Figure 3-15 Classless Inter-Domain Routing: IP supernetting example

This process of combining multiple networks into a single entry is referred to as *supernetting*. Routing is based on network masks that are shorter than the natural network mask of an IP address. This contrasts with subnetting (see 3.1.2, “IP subnets” on page 72) where the subnet masks are longer than the natural network mask.

The current Internet address allocation policies and the assumptions on which those policies were based are described in RFC 1518. They can be summarized as follows:

- IP address assignment reflects the physical topology of the network and not the organizational topology. Wherever organizational and administrative boundaries do not match the network topology, they should *not* be used for the assignment of IP addresses.
- In general, network topology will closely follow continental and national boundaries. Therefore, IP addresses should be assigned on this basis.

- There will be a relatively small set of networks that carry a large amount of traffic between routing domains. These networks will be interconnected in a non-hierarchical way that crosses national boundaries. These networks are referred to as *transit routing domains (TRDs)* Each TRD will have a unique IP prefix. TRDs will not be organized in a hierarchical way when there is no appropriate hierarchy. However, whenever a TRD is wholly within a continental boundary, its IP prefix should be an extension of the continental IP prefix.
- There will be many organizations that have attachments to other organizations that are for the private use of those two organizations. The attachments do not carry traffic intended for other domains (transit traffic). Such private connections do not have a significant effect on the routing topology and can be ignored.
- The great majority of routing domains will be single-homed. That is, they will be attached to a single TRD. They should be assigned addresses that begin with that TRD's IP prefix. All of the addresses for all single-homed domains attached to a TRD can therefore be aggregated into a single routing table entry for all domains outside that TRD.
- There are a number of address assignment schemes that can be used for multihomed domains. These include:
  - The use of a single IP prefix for the domain. External routers must have an entry for the organization that lies partly or wholly outside the normal hierarchy. Where a domain is multihomed, but all of the attached TRDs themselves are topologically nearby, it is appropriate for the domain's IP prefix to include those bits common to all of the attached TRDs. For example, if all of the TRDs were wholly within the United States, an IP prefix implying an exclusively North American domain is appropriate.
  - The use of one IP prefix for each attached TRD with hosts in the domain having IP addresses containing the IP prefix of the most appropriate TRD. The organization appears to be a set of routing domains.
  - Assigning an IP prefix from one of the attached TRDs. This TRD becomes a default TRD for the domain but other domains can explicitly route by one of the alternative TRDs.
  - The use of IP prefixes to refer to sets of multihomed domains having the TRD attachments. For example, there can be an IP prefix to refer to single-homed domains attached to network A, one to refer to single-homed domains attached to network B, and one to refer to dual-homed domains attached to networks A and B.

  Each of these has various advantages, disadvantages, and side effects. For example, the first approach tends to result in inbound traffic entering the target domain closer to the sending host than the second approach.

Therefore, a larger proportion of the network costs are incurred by the receiving organization.

Because multihomed domains vary greatly in character, none of the these schemes is suitable for every domain. There is no single policy that is best. RFC 1518 does not specify any rules for choosing between them.

CIDR implementation

The implementation of CIDR in the Internet is primarily based on Border Gateway Protocol Version 4 (see 5.9, “Border Gateway Protocol (BGP)” on page 215). The implementation strategy, described in RFC 1520, involves a staged process through the routing hierarchy beginning with backbone routers. Network service providers are divided into four types:

- Type 1: Those providers that cannot employ any default inter-domain routing.
- Type 2: Those providers that use default inter-domain routing but require explicit routes for a substantial proportion of the assigned IP network numbers.
- Type 3: Those providers that use default inter-domain routing and supplement it with a small number of explicit routes.
- Type 4: Those providers that perform inter-domain routing using only default routes.

The CIDR implementation began with the Type 1 network providers, then the Type 2, and finally the Type 3 providers. CIDR has already been widely deployed in the backbone and more than 190,000 class-based routes have been replaced by approximately 92,000 CIDR-based routes (through unique announced aggregates).

3.1.9 IP datagram

The unit of transfer in an IP network is called an IP datagram. It consists of an IP header and data relevant to higher-level protocols. See Figure 3-16 for details.

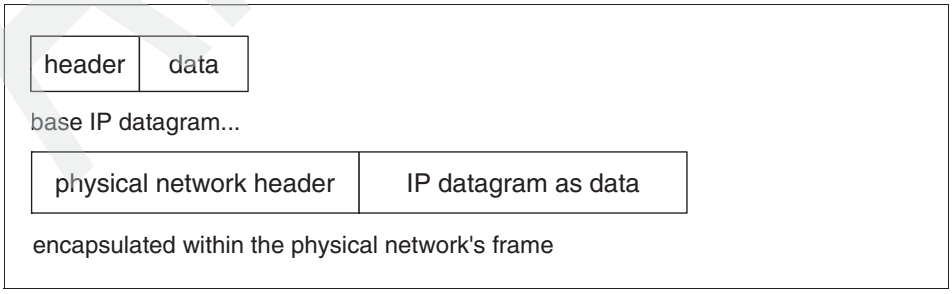


Figure 3-16 IP: Format of a base IP datagram

IP can provide fragmentation and reassembly of datagrams. The maximum length of an IP datagram is 65,535 octets. All IP hosts must support 576 octets datagrams without fragmentation.

Fragments of a datagram each have a header. The header is copied from the original datagram. A fragment is treated as a normal IP datagrams while being transported to their destination. However, if one of the fragments gets lost, the complete datagram is considered lost. Because IP does not provide any acknowledgment mechanism, the remaining fragments are discarded by the destination host.

IP datagram format

The IP datagram header has a minimum length of 20 octets, as illustrated in Figure 3-17.

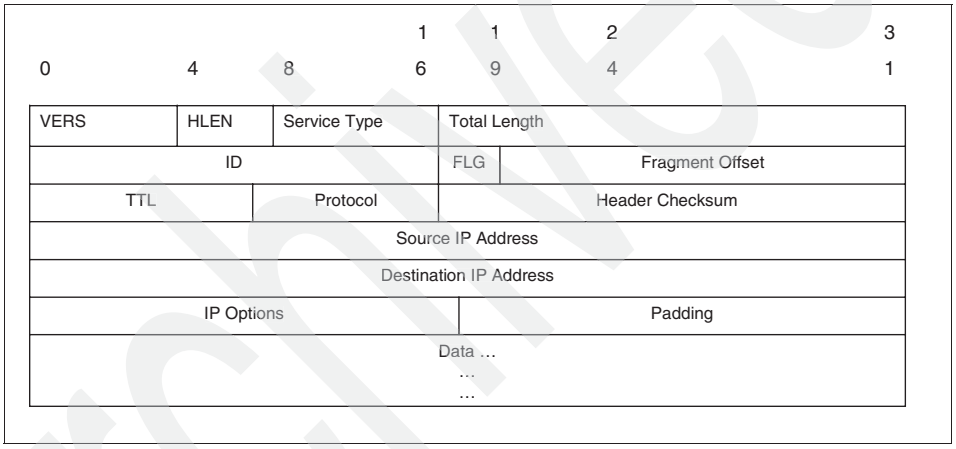


Figure 3-17 IP: Format of an IP datagram header

Where:

- VERS: The field contains the IP protocol version. The current version is 4. Version 5 is an experimental version. Version 6 is the version for IPv6 (see 9.2, “The IPv6 header format” on page 330).
- HLEN: The length of the IP header counted in 32-bit quantities. This does not include the data field.

Archived

- Service Type: The service type is an indication of the quality of service requested for this IP datagram. This field contains the information illustrated in Figure 3-18.

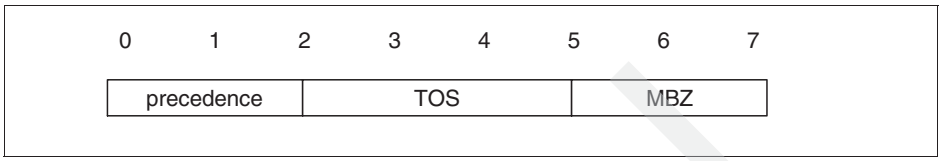


Figure 3-18 IP: Service type

Where:

  - Precedence: This field specifies the nature and priority of the datagram:
    - 000: Routine
    - 001: Priority
    - 010: Immediate
    - 011: Flash
    - 100: Flash override
    - 101: Critical
    - 110: Internetwork control
    - 111: Network control
  - TOS: Specifies the type of service value:
    - 1000: Minimize delay
    - 0100: Maximize throughput
    - 0010: Maximize reliability
    - 0001: Minimize monetary cost
    - 0000: Normal service

    A detailed description of the type of service is in the RFC 1349 (refer to 8.1, “Why QoS?” on page 288).
  - MBZ: Reserved for future use.
- Total Length: The total length of the datagram, header and data.
- Identification: A unique number assigned by the sender to aid in reassembling a fragmented datagram. Each fragment of a datagram has the same identification number.
- Flags: This field contains control flags illustrated in Figure 3-19.

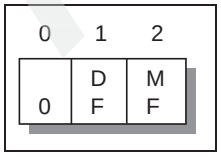


Figure 3-19 IP: Flags

Where:

- 0: Reserved, must be zero.
- DF (Do not Fragment): 0 means allow fragmentation; 1 means do not allow fragmentation.
- MF (More Fragments): 0 means that this is the last fragment of the datagram; 1 means that additional fragments will follow.

- Fragment Offset: This is used to aid the reassembly of the full datagram. The value in this field contains the number of 64-bit segments (header bytes are not counted) contained in earlier fragments. If this is the first (or only) fragment, this field contains a value of zero.
- Time to Live: This field specifies the time (in seconds) the datagram is allowed to travel. Theoretically, each router processing this datagram is supposed to subtract its processing time from this field. In practise, a router processes the datagram in less than 1 second. Therefore, the router subtracts one from the value in this field. The TTL becomes a hop-count metric rather than a time metric. When the value reaches zero, it is assumed that this datagram has been traveling in a closed loop and is discarded. The initial value should be set by the higher-level protocol that creates the datagram.
- Protocol Number: This field indicates the higher-level protocol to which IP should deliver the data in this datagram. These include:
  - 0: Reserved
  - 1: Internet Control Message Protocol (ICMP)
  - 2: Internet Group Management Protocol (IGMP)
  - 3: Gateway-to-Gateway Protocol (GGP)
  - 4: IP (IP encapsulation)
  - 5: Stream
  - 6: Transmission Control Protocol (TCP)
  - 8: Exterior Gateway Protocol (EGP)
  - 9: Interior Gateway Protocol (IGP)
  - 17: User Datagram Protocol (UDP)
  - 41:Simple Internet Protocol (SIP)
  - 50: SIPP Encap Security Payload (ESP)
  - 51: SIPP Authentication Header (AH)
  - 89: Open Shortest Path First (OSPF) IGP

  The complete list is in STD 2 – Assigned Internet Numbers.
- Header Checksum: This field is a checksum for the information contained in the header. If the header checksum does not match the contents, the datagram is discarded.
- Source IP Address: The 32-bit IP address of the host sending this datagram.

- Destination IP Address: The 32-bit IP address of the destination host for this datagram.
- Options: An IP implementation is not required to be capable of generating options in a datagram. However, all IP implementations are required to be able to process datagrams containing options. The Options field is variable in length (there can be zero or more options). There are two option formats. The format for each is dependent on the value of the option number found in the first octet:
  - A type octet alone is illustrated in Figure 3-20.

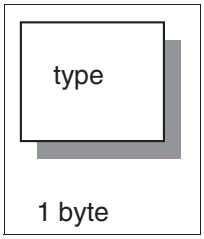


Figure 3-20 IP: A type byte

  - A type octet, a length octet, and one or more option data octets, as illustrated in Figure 3-21.

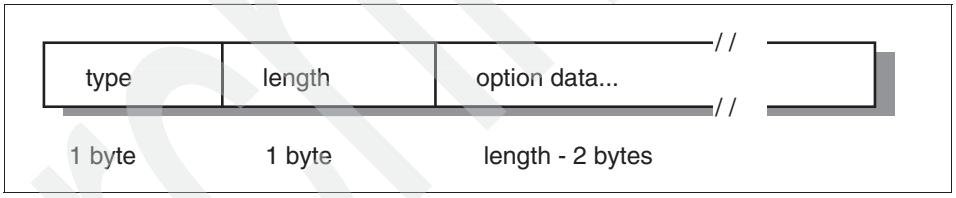


Figure 3-21 IP: A type byte, a length byte, and one or more option data bytes

The type byte has the same structure in both cases, as illustrated in Figure 3-22.

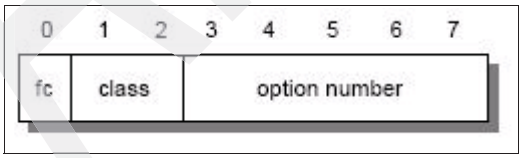


Figure 3-22 IP: The type byte structure

Where:

- fc (Flag copy): This field indicates whether (1) or not (0) the option field is copied when the datagram is fragmented.
- class: The option class is a 2-bit unsigned integer:
  - 0: Control
  - 1: Reserved
  - 2: Debugging and measurement
  - 3: Reserved
- option number: The option number is a 5-bit unsigned integer:
  - 0: End of option list. It has a class of 0, the fc bit is set to zero, and it has no length byte or data. That is, the option list is terminated by a X'00' byte. It is only required if the IP header length (which is a multiple of 4 bytes) does not match the actual length of the options.
  - 1: No operation. It has a class of 0, the fc bit is not set, and there is no length byte or data. That is, a X'01' byte is a NOP. It can be used to align fields in the datagram.
  - 2: Security. It has a class of 0, the fc bit is set, and there is a length byte with a value of 11 and 8 bytes of data). It is used for security information needed by U.S. Department of Defense requirements.
  - 3: Loose source routing. It has a class of 0, the fc bit is set, and there is a variable length data field. We discuss this option in more detail later.
  - 4: Internet time stamp. It has a class of 2, the fc bit is not set, and there is a variable length data field. The total length can be up to 40 bytes. We discuss this option in more detail later.
  - 7: Record route. It has a class of 0, the fc bit is not set, and there is a variable length data field. We discuss this option in more detail later.
  - 8: Stream ID. It has a class of 0, the fc bit is set, and there is a length byte with a value of 4 and one data byte. It is used with the SATNET system.
  - 9: Strict source routing. It has a class of 0, the fc bit is set, and there is a variable length data field. We discuss this option in more detail later.
- length: This field counts the length (in octets) of the option, including the type and length fields.
- option data: This field contains data relevant to the specific option.

- Padding: If an option is used, the datagram is padded with all-zero octets up to the next 32-bit boundary.
- Data: The data contained in the datagram. It is passed to the higher-level protocol specified in the protocol field.

Fragmentation

When an IP datagram travels from one host to another, it can pass through different physical networks. Each physical network has a maximum frame size. This is called the *maximum transmission unit* (MTU). It limits the length of a datagram that can be placed in one physical frame.

IP implements a process to fragment datagrams exceeding the MTU. The process creates a set of datagrams within the maximum size. The receiving host reassembles the original datagram. IP requires that each link support a minimum MTU of 68 octets. This is the sum of the maximum IP header length (60 octets) and the minimum possible length of data in a non-final fragment (8 octets). If any network provides a lower value than this, fragmentation and reassembly must be implemented in the network interface layer. This must be transparent to IP. IP implementations are not required to handle unfragmented datagrams larger than 576 bytes. In practice, most implementations will accommodate larger values.

An unfragmented datagram has an all-zero fragmentation information field. That is, the more fragments flag bit is zero and the fragment offset is zero. The following steps fragment the datagram:

1. The DF flag bit is checked to see if fragmentation is allowed. If the bit is set, the datagram will be discarded and an ICMP error returned to the originator.
2. Based on the MTU value, the data field is split into two or more parts. All newly created data portions must have a length that is a multiple of 8 octets, with the exception of the last data portion.
3. Each data portion is placed in an IP datagram. The headers of these datagrams are minor modifications of the original:
   - The more fragments flag bit is set in all fragments except the last.
   - The fragment offset field in each is set to the location this data portion occupied in the original datagram, relative to the beginning of the original unfragmented datagram. The offset is measured in 8-octet units.
   - If options were included in the original datagram, the high order bit of the option type byte determines if this information is copied to all fragment datagrams or only the first datagram. For example, source route options are copied in all fragments.
   - The header length field of the new datagram is set.
   - The total length field of the new datagram is set.
   - The header checksum field is re-calculated.
4. Each of these fragmented datagrams is now forwarded as a normal IP datagram. IP handles each fragment independently. The fragments can traverse different routers to the intended destination. They can be subject to further fragmentation if they pass through networks specifying a smaller MTU.

Archived

At the destination host, the data is reassembled into the original datagram. The identification field set by the sending host is used together with the source and destination IP addresses in the datagram. Fragmentation does not alter this field.

In order to reassemble the fragments, the receiving host allocates a storage buffer when the first fragment arrives. The host also starts a timer. When subsequent fragments of the datagram arrive, the data is copied into the buffer storage at the location indicated by the fragment offset field. When all fragments have arrived, the complete original unfragmented datagram is restored. Processing continues as for unfragmented datagrams.

If the timer is exceeded and fragments remain outstanding, the datagram is discarded. The initial value of this timer is called the IP datagram time to live (TTL) value. It is implementation-dependent. Some implementations allow it to be configured.

The **`netstat`** command can be used on some IP hosts to list the details of fragmentation.

IP datagram routing options

The IP datagram Options field provides two methods for the originator of an IP datagram to explicitly provide routing information. It also provides a method for an IP datagram to determine the route that it travels.

Loose source routing

The loose source routing option, also called the loose source and record route (LSRR) option, provides a means for the source of an IP datagram to supply explicit routing information. This information is used by the routers when forwarding the datagram to the destination. It is also used to record the route, as illustrated in Figure 3-23.

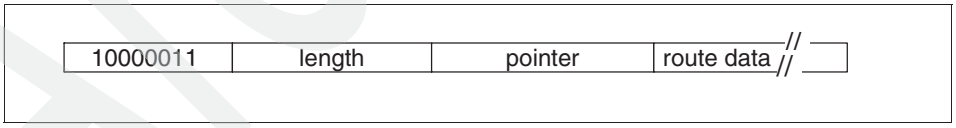


Figure 3-23 IP: Loose source routing option

The fields of this header include:

1000011(decimal 131) This is the value of the option type octet for loose source routing.

Length This field contains the length of this option field, including the type and length fields.

Archive

Pointer This field points to the option data at the next IP address to be processed. It is counted relative to the beginning of the option, so its minimum value is four. If the pointer is greater than the length of the option, the end of the source route is reached and further routing is to be based on the destination IP address (as for datagrams without this option).

Route data This field contains a series of 32-bit IP addresses.

When a datagram arrives at its destination and the source route is not empty (pointer < length) the receiving host:

1. Takes the next IP address in the route data field (the one indicated by the pointer field) and puts it in the destination IP address field of the datagram.
2. Puts the local IP address in the source list at the location pointed to by the pointer field. The IP address for this is the local IP address corresponding to the network on which the datagram will be forwarded. (Routers are attached to multiple physical networks and thus have multiple IP addresses.)
3. Increments the pointer by 4.
4. Transmits the datagram to the new destination IP address.

This procedure ensures that the return route is recorded in the route data (in reverse order) so that the final recipient uses this data to construct a loose source route in the reverse direction. This is a *loose* source route because the forwarding router is allowed to use any route and any number of intermediate routers to reach the next address in the route.

Strict source routing

The strict source routing option, also called the strict source and record route (SSRR) option, uses the same principle as loose source routing except the intermediate router *must* send the datagram to the next IP address in the source route through a directly connected network. It cannot use an intermediate router. If this cannot be done, an ICMP Destination Unreachable error message is issued. Figure 3-24 gives an overview of the SSRR option.

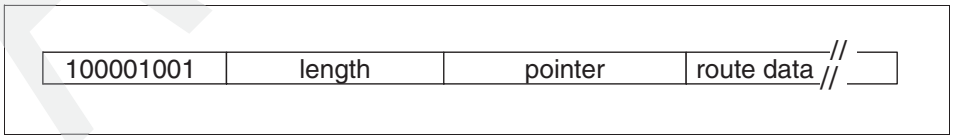


Figure 3-24 IP: Strict source routing option

Where:

1001001 (Decimal 137) The value of the option type byte for strict source routing.

Length This information is described in “Loose source routing” on page 105.

Pointer This information is described in “Loose source routing” on page 105.

Route data A series of 32-bit IP addresses.

Record route

This option provides a means to record the route traversed by an IP datagram. It functions similarly to the source routing option. However, this option provides an empty routing data field. This field is filled in as the datagram traverses the network. Sufficient space for this routing information must be provided by the source host. If the data field is filled before the datagram reaches its destination, the datagram is forwarded with no further recording of the route. Figure 3-25 gives an overview of the record route option.

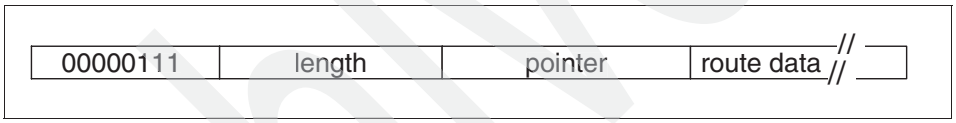


Figure 3-25 IP: Record route option

Where:

0000111 (Decimal 7) The value of the option type byte for record route

Length This information is described in “Loose source routing” on page 105.

Pointer This information is described in “Loose source routing” on page 105.

Route data A series of 32-bit IP addresses.

Internet time stamp

A time stamp is an option forcing some (or all) of the routers along the route to the destination to put a time stamp in the option data. The time stamps are measured in seconds and can be used for debugging purposes. They cannot be used for performance measurement for two reasons:

- Because most IP datagrams are forwarded in less than one second, the time stamps are not precise.

Archived

- Because IP routers are not required to have synchronized clocks, they may not be accurate.

Figure 3-26 gives an overview of the Internet time stamp option.

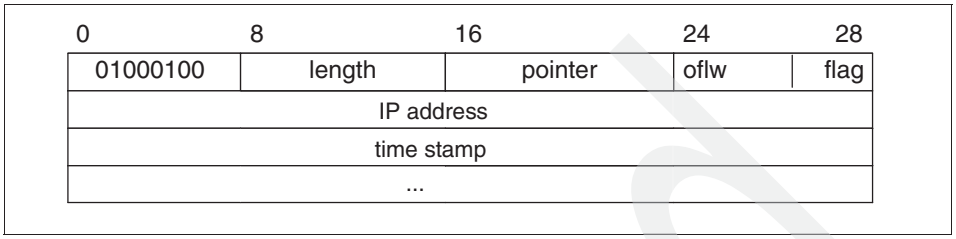


Figure 3-26 IP: Internet time stamp option

Where:

01000100 (Decimal 68) This field is the value of the option type for the internet time stamp option.

Length This field contains the total length of this option, including the type and length fields.

Pointer This field points to the next time stamp to be processed (first free time stamp).

Oflw (overflow) This field contains the number of devices that cannot register time stamps due to a lack of space in the data field.

Flag This field is a 4-bit value that indicates how time stamps are to be registered:

- **0** Time stamps only, stored in consecutive 32-bit words.
- **1** Each time stamp is preceded by the IP address of the registering device.
- **2** The IP address fields are prespecified; an IP device only registers when it finds its own address in the list.

Time stamp A 32-bit time stamp recorded in milliseconds since midnight UT (GMT).

The originating host must compose this option with a sufficient data area to hold all the time stamps. If the time stamp area becomes full, no further time stamps are added.

Archived

3.2 Internet Control Message Protocol (ICMP)

ICMP is a standard protocol with STD number 5. That standard also includes IP (see 3.1, “Internet Protocol (IP)” on page 68) and IGMP (see 6.2, “Internet Group Management Protocol (IGMP)” on page 241). Its status is required. It is described in RFC 792 with updates in RFC 950. ICMPv6 used for IPv6 is discussed in 9.3, “Internet Control Message Protocol Version 6 (ICMPv6)” on page 352.

Path MTU Discovery is a draft standard protocol with a status of elective. It is described in RFC 1191.

ICMP Router Discovery is a proposed standard protocol with a status of elective. It is described in RFC 1256.

When a router or a destination host must inform the source host about errors in datagram processing, it uses the Internet Control Message Protocol (ICMP). ICMP can be characterized as follows:

- ICMP uses IP as though ICMP were a higher-level protocol (that is, ICMP messages are encapsulated in IP datagrams). However, ICMP is an integral part of IP and must be implemented by every IP module.
- ICMP is used to report errors, *not* to make IP reliable. Datagrams can still be undelivered without any report on their loss. Reliability must be implemented by the higher-level protocols using IP services.
- ICMP cannot be used to report errors with ICMP messages. This avoids infinite repetitions. ICMP responses are sent in response to ICMP query messages (ICMP types 0, 8, 9, 10, and 13 through 18).
- For fragmented datagrams, ICMP messages are only sent about errors with the first fragment. That is, ICMP messages never refer to an IP datagram with a non-zero fragment offset field.
- ICMP messages are never sent in response to datagrams with a broadcast or a multicast destination address.
- ICMP messages are never sent in response to a datagram that does not have a source IP address representing a unique host. That is, the source address cannot be zero, a loopback address, a broadcast address, or a multicast address.
- RFC 792 states that ICMP messages *can* be generated to report IP datagram processing errors. However, this is not required. In practice, routers will almost always generate ICMP messages for errors. For destination hosts, ICMP message generation is implementation dependent.

3.2.1 ICMP messages

ICMP messages are described in RFC 792 and RFC 950, belong to STD 5, and are mandatory.

ICMP messages are sent in IP datagrams. The IP header has a protocol number of 1 (ICMP) and a type of service of zero (routine). The IP data field contains the ICMP message shown in Figure 3-27.

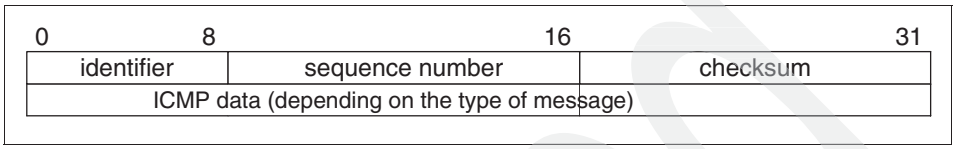


Figure 3-27 ICMP: Message format

The message contains the following components:

Type Specifies the type of the message:

- **0** Echo reply
- **3** Destination unreachable
- **4** Source quench
- **5** Redirect
- **8** Echo
- **9** Router advertisement
- **10** Router solicitation
- **11** Time exceeded
- **12** Parameter problem
- **13** Time stamp request
- **14** Time stamp reply
- **17** Address mask request
- **18** Address mask reply
- **30** Traceroute
- **37** Domain name request)
- **38** Domain name reply)

The following RFCs are required to be mentioned for some of the ICMP message types: RFC 1256, RFC 1393, and RFC 1788.

Code Contains the error code for the datagram reported by this ICMP message. The interpretation is dependent on the message type.

Checksum Contains the checksum for the ICMP message starting with the ICMP Type field. If the checksum does not match the contents, the datagram is discarded.

Archived

Data Contains information for this ICMP message. Typically, it will contain the portion of the original IP message for which this ICMP message was generated.

Each of the ICMP messages is described individually.

Echo (8) and Echo Reply (0)

Echo is used to detect if another host is active in the network. It is used by the Ping command (refer to “Ping” on page 117). The sender initializes the identifier, sequence number, and data field. The datagram is then sent to the destination host. The recipient changes the type to Echo Reply and returns the datagram to the sender. See Figure 3-28 for more details.

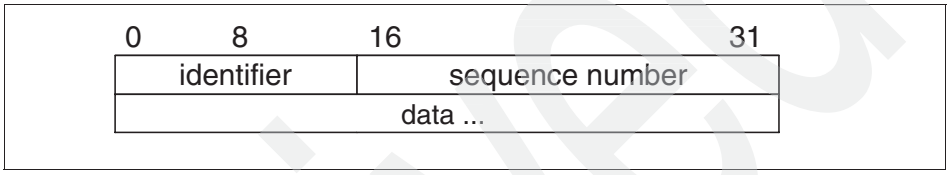


Figure 3-28 Echo and Echo Reply

Destination Unreachable (3)

If this message is received from an intermediate router, it means that the router regards the destination IP address as unreachable.

If this message is received from the destination host, it means that either the protocol specified in the protocol number field of the original datagram is not active or the specified port is inactive. (Refer to 4.2, “User Datagram Protocol (UDP)” on page 146 for additional information regarding ports.) See Figure 3-29 for more details.

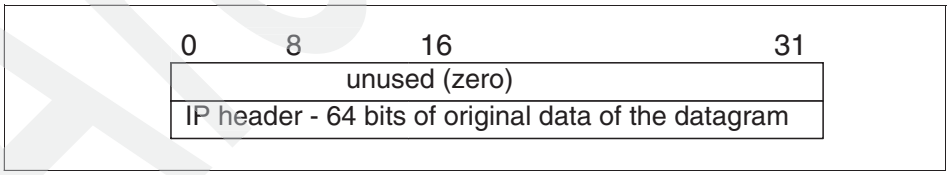


Figure 3-29 ICMP: Destination Unreachable

The ICMP header code field contains one of the following values:

0 Network unreachable
1 Host unreachable
2 Protocol unreachable
3 Port unreachable
4 Fragmentation needed but the Do Not Fragment bit was set

5 Source route failed
6 Destination network unknown
7 Destination host unknown
8 Source host isolated (obsolete)
9 Destination network administratively prohibited
10 Destination host administratively prohibited
11 Network unreachable for this type of service
12 Host unreachable for this type of service
13 Communication administratively prohibited by filtering
14 Host precedence violation
15 Precedence cutoff in effect

These are detailed in RFC 792, RFC 1812 updated by RFC 2644, RFC 1122, updated by RFC 4379, and forms part of STD 3 – Host Requirements.

If a router implements the Path MTU Discovery protocol, the format of the destination unreachable message is changed for code 4. This includes the MTU of the link that did not accept the datagram. See Figure 3-30 for more information.

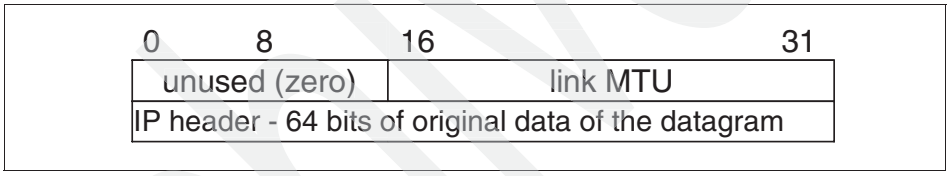


Figure 3-30 ICMP: Fragmentation required with link MTU

Source Quench (4)

If this message is received from an intermediate router, it means that the router did not have the buffer space needed to queue the datagram.

If this message is received from the destination host, it means that the incoming datagrams are arriving too quickly to be processed.

The ICMP header code field is always zero.

See Figure 3-31 for more details.

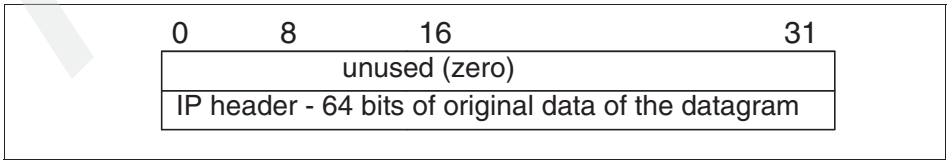


Figure 3-31 ICMP: Source Quench

Redirect (5)

If this message is received from an intermediate router, it means that the host should send future datagrams for the network to the router whose IP address is specified in the ICMP message. This preferred router will always be on the same subnet as the host that sent the datagram and the router that returned the IP datagram. The router forwards the datagram to its next hop destination. This message will not be sent if the IP datagram contains a source route.

The ICMP header code field will have one of the following values:

0 Network redirect
1 Host redirect
2 Network redirect for this type of service
3 Host redirect for this type of service

See Figure 3-32 for more details.

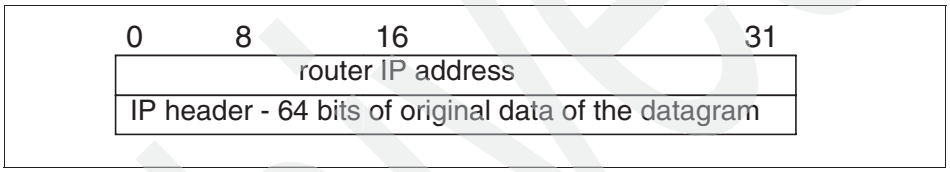


Figure 3-32 ICMP: Redirect

Router Advertisement (9) and Router Solicitation (10)

ICMP messages 9 and 10 are optional. They are described in RFC 1256, which is elective. See Figure 3-33 and Figure 3-34 on page 114 for details.

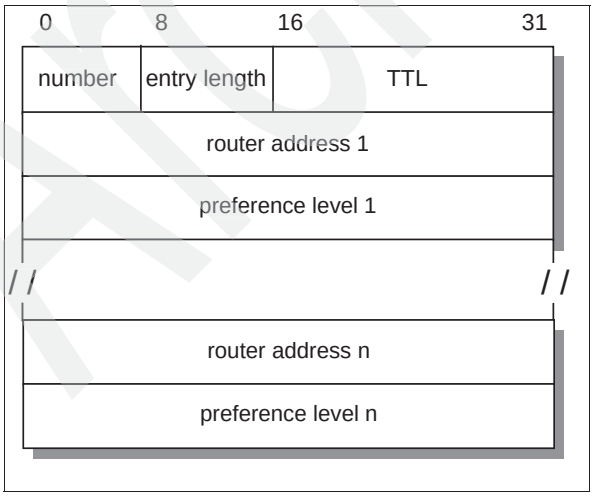


Figure 3-33 ICMP: Router Advertisement

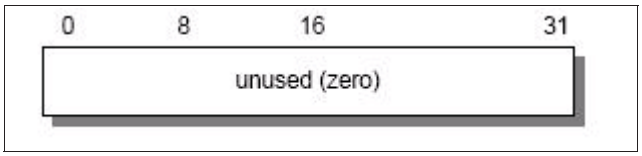


Figure 3-34 ICMP: Router Solicitation

Where:

Number The number of entries in the message.

Entry length The length of an entry in 32-bit units. This is 2 (32 bits for the IP address and 32 bits for the preference value).

TTL The number of seconds that an entry will be considered valid.

Router address One of the sender's IP addresses.

Preference level A signed 32-bit level indicating the preference to be assigned to this address when selecting a default router. Each router on a subnet is responsible for advertising its own preference level. Larger values imply higher preference; smaller values imply lower. The default is zero, which is in the middle of the possible range. A value of X'80000000' (-231) indicates the router should never be used as a default router.

The ICMP header code field is zero for both of these messages.

These two messages are used if a host or a router supports the router discovery protocol. Routers periodically advertise their IP addresses on those subnets where they are configured to do so. Advertisements are made on the all-systems multicast address (224.0.0.1) or the limited broadcast address (255.255.255.255). The default behavior is to send advertisements every 10 minutes with a TTL value of 1800 (30 minutes). Routers also reply to solicitation messages they receive. They might reply directly to the soliciting host, or they might wait a short random interval and reply with a multicast.

Hosts can send solicitation messages. Solicitation messages are sent to the all-routers multicast address (224.0.0.2) or the limited broadcast address (255.255.255.255). Typically, three solicitation messages are sent at 3-second intervals. Alternatively, a host can wait for periodic advertisements. Each time a host receives an advertisement with a higher preference value, it updates its default router. The host also sets the TTL timer for the new entry to match the value in the advertisement. When the host receives a new advertisement for its current default router, it resets the TTL value to that in the new advertisement.

This process also provides a mechanism for routers to declare themselves unavailable. They send an advertisement with a TTL value of zero.

Time Exceeded (11)

If this message is received from an intermediate router, it means that the time to live field of an IP datagram has expired.

If this message is received from the destination host, it means that the IP fragment reassembly time to live timer has expired while the host is waiting for a fragment of the datagram. The ICMP header code field can have the one of the following values:

0 Transit TTL exceeded
1 Reassembly TTL exceeded

See Figure 3-35 for more details.

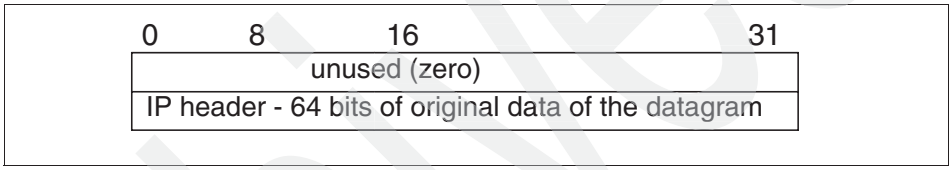


Figure 3-35 ICMP: Time Exceeded

Parameter Problem (12)

This message indicates that a problem was encountered during processing of the IP header parameters. The pointer field indicates the octet in the original IP datagram where the problem was encountered. The ICMP header code field can have the one of the following values:

0 Unspecified error
1 Required option missing

See Figure 3-36 for more details.

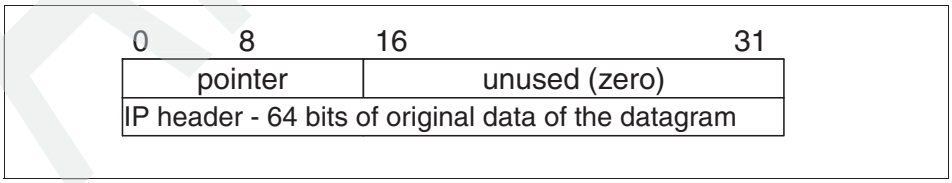


Figure 3-36 ICMP: Parameter Problem

Archived

Timestamp Request (13) and Timestamp Reply (14)

These two messages are for debugging and performance measurements. They are not used for clock synchronization.

The sender initializes the identifier and sequence number (which is used if multiple time stamp requests are sent), sets the originate time stamp, and sends the datagram to the recipient. The receiving host fills in the receive and transmit time stamps, changes the type to time stamp reply, and returns it to the original sender. The datagram has two time stamps if there is a perceptible time difference between the receipt and transmit times. In practice, most implementations perform the two (receipt and reply) in one operation. This sets the two time stamps to the same value. Time stamps are the number of milliseconds elapsed since midnight UT (GMT).

See Figure 3-37 for details.

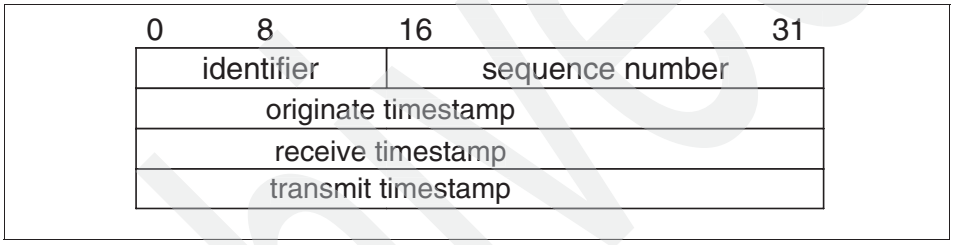


Figure 3-37 ICMP: Timestamp Request and Timestamp Reply

Address Mask Request (17) and Address Mask Reply (18)

An address mask request is used by a host to determine the subnet mask used on an attached network. Most hosts are configured with their subnet mask or masks. However some, such as diskless workstations, must obtain this information from a server. A host uses RARP (see 3.5, “Reverse Address Resolution Protocol (RARP)” on page 124) to obtain its IP address. To obtain a subnet mask, the host broadcasts an address mask request. Any host in the network that has been configured to send address mask replies will fill in the subnet mask, convert the packet to an address mask reply, and return it to the sender. The ICMP header code field is zero.

See Figure 3-38 on page 117 for more details.

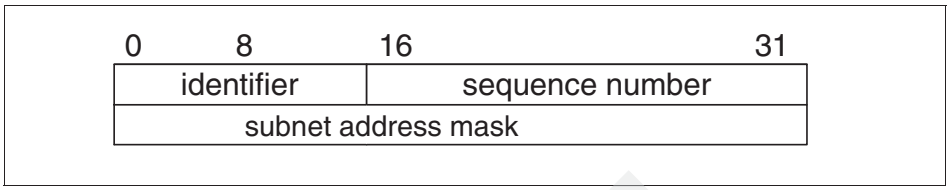


Figure 3-38 ICMP: Address Mask Request and Reply

3.2.2 ICMP applications

There are two simple and widely used applications based on ICMP: Ping and Traceroute. Ping uses the ICMP Echo and Echo Reply messages to determine whether a host is reachable. Traceroute sends IP datagrams with low TTL values so that they expire en route to a destination. It uses the resulting ICMP Time Exceeded messages to determine where in the internet the datagrams expired and pieces together a view of the route to a host. We discuss these applications in the following sections.

Ping

Ping is the simplest of all TCP/IP applications. It sends IP datagrams to a specified destination host and measures the round trip time to receive a response. The word *ping*, which is used as a noun and a verb, is taken from the sonar operation to locate an underwater object. It is also an abbreviation for *Packet InterNet Groper*.

Generally, the first test of reachability for a host is to attempt to ping it. If you can successfully ping a host, other applications such as Telnet or FTP should be able to reach that host. However with the advent of security measures on the Internet, particularly firewalls (see 22.3, “Firewalls” on page 794), which control access to networks by application protocol or port number, or both, this is no longer necessarily true. The ICMP protocol can be restricted on the firewall and therefore the host is unable to be successfully pinged.

The syntax that is used in different implementations of ping varies from platform to platform. A common format for using the ping command is:

```
ping host
```

Where host is the destination, either a symbolic name or an IP address.

Most platforms allow you to specify the following values:

Size The size of the data portion of the packet.
Packets The number of packets to send.
Count The number of echo requests to send.

Record routes	Record the route per count hop.
Time stamp	Time stamp each count hop.
Endless ping	Ping until manually stopped.
Resolve address	Resolve the host address to the host name.
Time to Live (TTL)	The time (in seconds) the datagram is allowed to travel.
Type of Service (TOS)	The type of internet service quality.
Host-list	Loose source route or strict source route of host lists.
Timeout	The timeout to wait for each reply.
No fragmentation	The fragment flag is not set.

Ping uses the ICMP Echo and Echo Reply messages (refer to “Echo (8) and Echo Reply (0)” on page 111). Because ICMP is required in every TCP/IP implementation, hosts do not require a separate server to respond to ping requests.

Ping is useful for verifying an IP installation. The following variations of the command each require the operation of an different portion of an IP installation:

- ping loopback: Verifies the operation of the base TCP/IP software.
- ping my-IP-address: Verifies whether the physical network device can be addressed.
- ping a-remote-IP-address: Verifies whether the network can be accessed.
- ping a-remote-host-name: Verifies the operation of the name server (or the flat namespace resolver, depending on the installation).

Traceroute

The Traceroute program is used to determine the route IP datagrams follow through the network.

Traceroute is based on ICMP and UDP. It sends an IP datagram with a TTL of 1 to the destination host. The first router decrements the TTL to 0, discards the datagram, and returns an ICMP Time Exceeded message to the source. In this way, the first router in the path is identified. This process is repeated with successively larger TTL values to identify the exact series of routers in the path to the destination host.

Traceroute sends UDP datagrams to the destination host. These datagrams reference a port number outside the standard range. When an ICMP Port Unreachable message is received, the source determines the destination host has been reached.

3.3 Internet Group Management Protocol (IGMP)

IGMP is a standard protocol with STD number 5. That standard also includes IP (see 3.1, “Internet Protocol (IP)” on page 68) and ICMP (see 3.2, “Internet Control Message Protocol (ICMP)” on page 109). Its status is recommended. It is described in RFC 1112 with updates in RFC 2236.

Similar to ICMP, the Internet Group Management Protocol (IGMP) is also an integral part of IP. It allows hosts to participate in IP multicasts. IGMP further provides routers with the capability to check if any hosts on a local subnet are interested in a particular multicast.

Refer to 6.2, “Internet Group Management Protocol (IGMP)” on page 241 for a detailed review of IGMP.

3.4 Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a network-specific standard protocol. The address resolution protocol is responsible for converting the higher-level protocol addresses (IP addresses) to physical network addresses. It is described in RFC 826.

3.4.1 ARP overview

On a single physical network, individual hosts are known in the network by their physical hardware address. Higher-level protocols address destination hosts in the form of a symbolic address (IP address in this case). When such a protocol wants to send a datagram to destination IP address w.x.y.z, the device driver does not understand this address.

Therefore, a module (ARP) is provided that will translate the IP address to the physical address of the destination host. It uses a lookup table (sometimes referred to as the *ARP cache*) to perform this translation.

When the address is not found in the ARP cache, a broadcast is sent out in the network with a special format called the *ARP request*. If one of the machines in the network recognizes its own IP address in the request, it will send an *ARP reply* back to the requesting host. The reply will contain the physical hardware address of the host and source route information (if the packet has crossed bridges on its path). Both this address and the source route information are stored in the ARP cache of the requesting host. All subsequent datagrams to this destination IP address can now be translated to a physical address, which is used by the device driver to send out the datagram in the network.

An exception to the rule is the asynchronous transfer mode (ATM) technology, where ARP cannot be implemented in the physical layer as described previously. Therefore, every host, upon initialization, must register with an ARP server in order to be able to resolve IP addresses to hardware addresses (also see 2.10, “Asynchronous transfer mode (ATM)” on page 47).

ARP was designed to be used on networks that support hardware broadcast. This means, for example, that ARP will not work on an X.25 network.

3.4.2 ARP detailed concept

ARP is used on IEEE 802 networks as well as on the older DIX Ethernet networks to map IP addresses to physical hardware addresses (see 2.1, “Ethernet and IEEE 802 local area networks (LANs)” on page 30). To do this, it is closely related to the device driver for that network. In fact, the ARP specifications in RFC 826 only describe its functionality, not its implementation. The implementation depends to a large extent on the device driver for a network type and they are usually coded together in the *adapter microcode*.

ARP packet generation

If an application wants to send data to a certain IP destination address, the IP routing mechanism first determines the IP address of the next hop of the packet (it can be the destination host itself, or a router) and the hardware device on which it should be sent. If it is an IEEE 802.3/4/5 network, the ARP module must be consulted to map the <protocol type, target protocol address> to a physical address.

The ARP module tries to find the address in this ARP cache. If it finds the matching pair, it gives the corresponding 48-bit physical address back to the caller (the device driver), which then transmits the packet. If it does not find the pair in its table, it *discards the packet* (the assumption is that a higher-level protocol will retransmit) and generates a network *broadcast* of an ARP request. See Figure 3-39 on page 121 for more details.

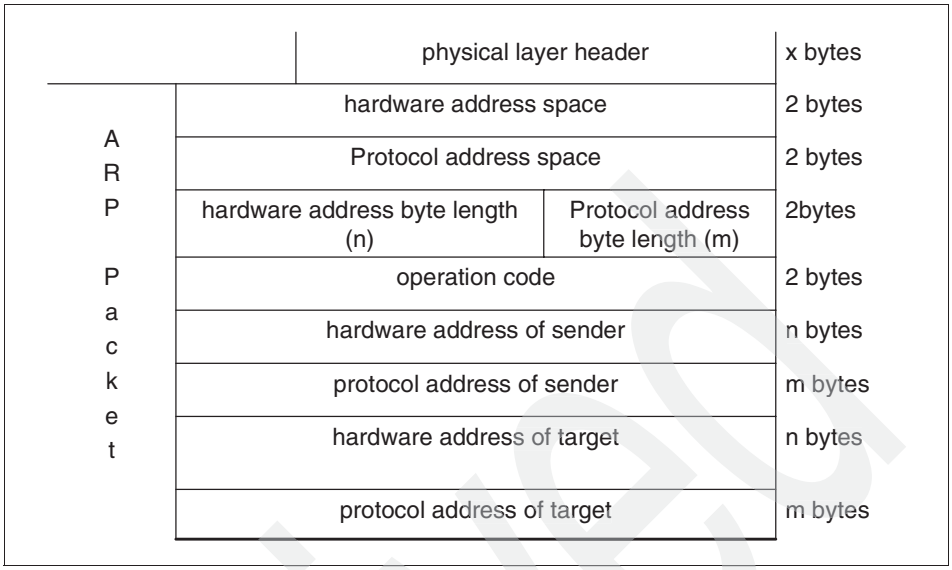


Figure 3-39 ARP: Request/reply packet

Where:

- Hardware address space: Specifies the type of hardware; examples are Ethernet or Packet Radio Net.
- Protocol address space: Specifies the type of protocol, same as the EtherType field in the IEEE 802 header (IP or ARP).
- Hardware address length: Specifies the length (in bytes) of the hardware addresses in this packet. For IEEE 802.3 and IEEE 802.5, this is 6.
- Protocol address length: Specifies the length (in bytes) of the protocol addresses in this packet. For IP, this is 4.
- Operation code: Specifies whether this is an ARP request (1) or reply (2).
- Source/target hardware address: Contains the physical network hardware addresses. For IEEE 802.3, these are 48-bit addresses.
- Source/target protocol address: Contains the protocol addresses. For TCP/IP, these are the 32-bit IP addresses.

For the ARP request packet, the target hardware address is the only undefined field in the packet.

ARP packet reception

When a host receives an ARP packet (either a broadcast request or a point-to-point reply), the receiving device driver passes the packet to the ARP module, which treats it as shown in Figure 3-40.

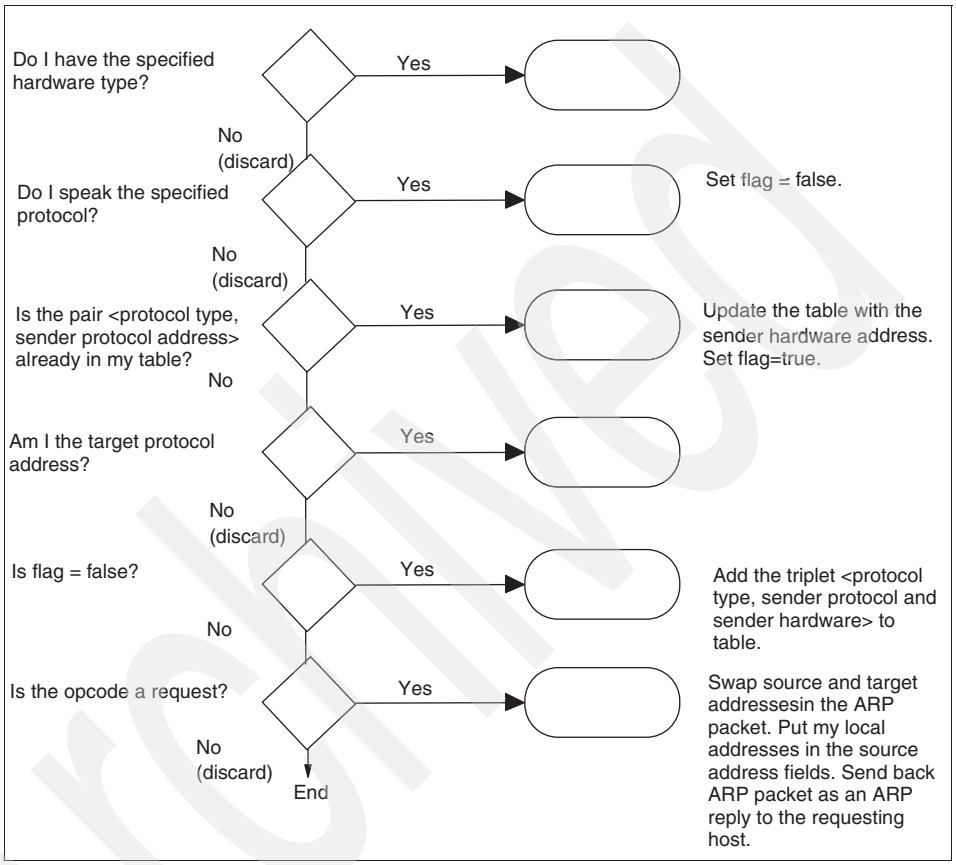


Figure 3-40 ARP: Packet reception

The requesting host will receive this ARP reply, and will follow the same algorithm to treat it. As a result of this, the triplet <protocol type, protocol address, hardware address> for the desired host will be added to its lookup table (ARP cache). The next time a higher-level protocol wants to send a packet to that host, the ARP module will find the target hardware address and the packet will be sent to that host.

Note that because the original ARP request was a broadcast in the network, all hosts on that network will have updated the sender's hardware address in their table (only if it was already in the table).

3.4.3 ARP and subnets

The ARP protocol remains unchanged in the presence of subnets. Remember that each IP datagram first goes through the IP routing algorithm. This algorithm selects the hardware device driver that should send out the packet. Only then, the ARP module associated with that device driver is consulted.

3.4.4 Proxy-ARP or transparent subnetting

Proxy-ARP is described in RFC 1027, which is a subset of the method proposed in RFC 925. It is another method to construct local subnets, without the need for a modification to the IP routing algorithm, but with modifications to the routers that interconnect the subnets.

Proxy-ARP concept

Consider one IP network that is divided into subnets and interconnected by routers. We use the “old” IP routing algorithm, which means that no host knows about the existence of multiple physical networks. Consider hosts A and B, which are on different physical networks within the same IP network, and a router R between the two subnetworks as illustrated in Figure 3-41.

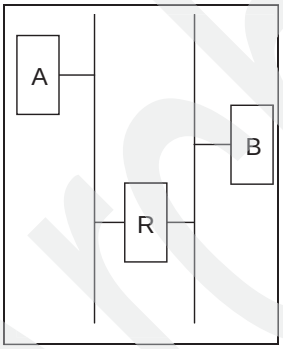


Figure 3-41 ARP: Hosts interconnected by a router

When host A wants to send an IP datagram to host B, it first has to determine the physical network address of host B through the use of the ARP protocol.

Because host A cannot differentiate between the physical networks, its IP routing algorithm thinks that host B is on the local physical network and sends out a broadcast ARP request. Host B does not receive this broadcast, but router R does. Router R understands subnets, that is, it runs the subnet version of the IP routing algorithm and it will be able to see that the destination of the ARP request (from the target protocol address field) is on another physical network. If router R's routing tables specify that the next hop to that other network is through a

different physical device, it will reply to the ARP as though it were host B, saying that the network address of host B is that of the router R itself.

Host A receives this ARP reply, puts it in its cache, and will send future IP packets for host B to the router R. The router will forward such packets to the correct subnet.

The result is transparent subnetting:

- Normal hosts (such as A and B) do not know about subnetting, so they use the “old” IP routing algorithm.
- The routers between subnets have to:
  - Use the subnet IP routing algorithm.
  - Use a modified ARP module, which can reply on behalf of other hosts.

See Figure 3-42 for more details.

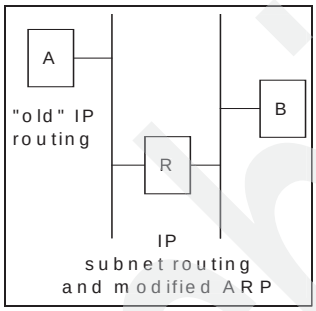


Figure 3-42 ARP: Proxy-ARP router

3.5 Reverse Address Resolution Protocol (RARP)

Reverse Address Resolution Protocol (RARP) is a network-specific standard protocol. It is described in RFC 903.

Some network hosts, such as diskless workstations, do not know their own IP address when they are booted. To determine their own IP address, they use a mechanism similar to ARP, but now the hardware address of the host is the known parameter and the IP address the queried parameter. It differs more fundamentally from ARP in the fact that a RARP server must exist in the network that maintains that a database of mappings from hardware address to protocol address must be preconfigured.

3.5.1 RARP concept

The reverse address resolution is performed the same way as the ARP address resolution. The same packet format (see Figure 3-39 on page 121) is used as for ARP.

An exception is the operation code field that now takes the following values:

3 For the RARP request
4 For the RARP reply

And of course, the physical header of the frame will now indicate RARP as the higher-level protocol (8035 hex) instead of ARP (0806 hex) or IP (0800 hex) in the EtherType field.

Some differences arise from the concept of RARP itself:

- ARP only assumes that every host knows the mapping between its own hardware address and protocol address. RARP requires one or more server hosts in the network to maintain a database of mappings between hardware addresses and protocol addresses so that they will be able to reply to requests from client hosts.
- Due to the size this database can take, part of the server function is usually implemented outside the adapter's microcode, with optionally a small cache in the microcode. The microcode part is then only responsible for reception and transmission of the RARP frames, the RARP mapping itself being taken care of by server software running as a normal process on the host machine.
- The nature of this database also requires some software to create and update the database manually.
- If there are multiple RARP servers in the network, the RARP requester only uses the first RARP reply received on its broadcast RARP request and discards the others.

3.6 Bootstrap Protocol (BOOTP)

The Bootstrap Protocol (BOOTP) enables a client workstation to initialize with a minimal IP stack and request its IP address, a gateway address, and the address of a name server from a BOOTP server. If BOOTP is to be used in your network, the server and client are usually on the same physical LAN segment. BOOTP can only be used across bridged segments when source-routing bridges are being used, or across subnets, if you have a router capable of BOOTP forwarding.

BOOTP is a draft standard protocol. Its status is recommended. The BOOTP specifications are in RFC 951, which has been updated by RFC1542 and RFC 2132.

There are also updates to BOOTP, some relating to interoperability with DHCP (see 3.7, “Dynamic Host Configuration Protocol (DHCP)” on page 130), described in RFC 1542, which updates RFC 951 and RFC 2132. The updates to BOOTP are draft standards with a status of elective and recommended, respectively.

The BOOTP protocol was originally developed as a mechanism to enable diskless hosts to be remotely booted over a network as workstations, routers, terminal concentrators, and so on. It allows a minimum IP protocol stack with no configuration information to obtain enough information to begin the process of downloading the necessary boot code. BOOTP does not define how the downloading is done, but this process typically uses TFTP (see also 14.2, “Trivial File Transfer Protocol (TFTP)” on page 529), as described in RFC 906. Although still widely used for this purpose by diskless hosts, BOOTP is also commonly used solely as a mechanism to deliver configuration information to a client that has not been manually configured.

The BOOTP process involves the following steps:

1. The client determines its own hardware address; this is normally in a ROM on the hardware.
2. A BOOTP client sends its hardware address in a UDP datagram to the server. Figure 3-43 on page 127 shows the full contents of this datagram. If the client knows its IP address or the address of the server, it should use them, but in general, BOOTP clients have no IP configuration data at all. If the client does not know its own IP address, it uses 0.0.0.0. If the client does not know the server's IP address, it uses the limited broadcast address (255.255.255.255). The UDP port number is 67.
3. The server receives the datagram and looks up the hardware address of the client in its configuration file, which contains the client's IP address. The server fills in the remaining fields in the UDP datagram and returns it to the client using UDP port 68. One of three methods can be used to do this:
   – If the client knows its own IP address (it was included in the BOOTP request), the server returns the datagram directly to this address. It is likely that the ARP cache in the server's protocol stack will not know the hardware address matching the IP address. ARP will be used to determine it as normal.
   – If the client does not know its own IP address (it was 0.0.0.0 in the BOOTP request), the server must concern itself with its own ARP cache.

– ARP on the server cannot be used to find the hardware address of the client because the client does not know its IP address and so cannot reply to an ARP request. This is called the “chicken and egg” problem. There are two possible solutions:
  - If the server has a mechanism for directly updating its own ARP cache without using ARP itself, it does so and then sends the datagram directly.
  - If the server cannot update its own ARP cache, it must send a broadcast reply.

4. When it receives the reply, the BOOTP client will record its own IP address (allowing it to respond to ARP requests) and begin the bootstrap process.

Figure 3-43 gives an overview of the BOOTP message format.

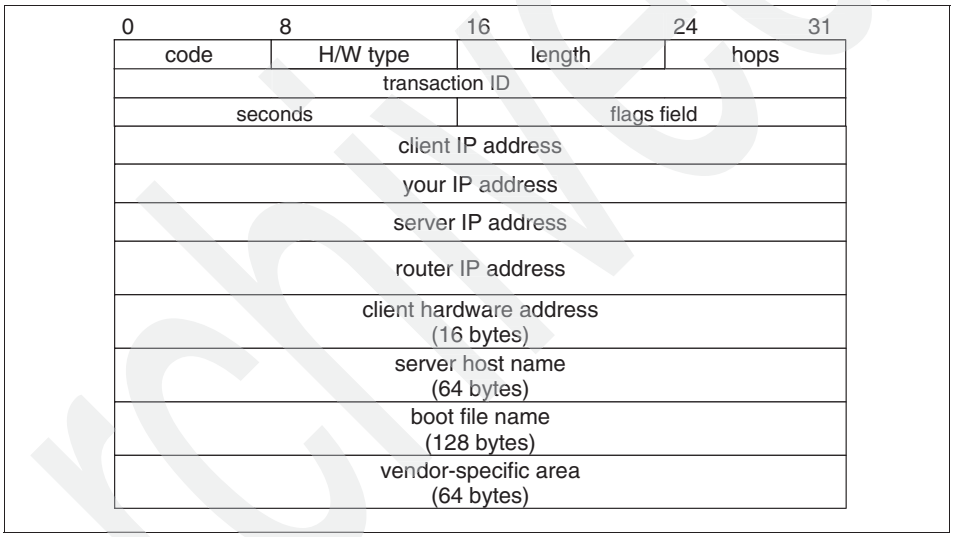


Figure 3-43 BOOTP message format

Where:

Code Indicates a request or a reply:

1 Request
2 Reply

H/W type The type of hardware, for example:

1 Ethernet
6 IEEE 802 Networks

Refer to STD 2 – Assigned Internet Numbers for a complete list.

Length Hardware address length in bytes. Ethernet and token ring both use 6, for example.

Hops The client sets this to 0.
It is incremented by a router that relays the request to another server and is used to identify loops. RFC 951 suggests that a value of 3 indicates a loop.

Transaction ID A random number used to match this boot request with the response it generates.

Seconds Set by the client. It is the elapsed time in seconds since the client started its boot process.

Flags field The most significant bit of the flags field is used as a broadcast flag. All other bits must be set to zero; they are reserved for future use. Normally, BOOTP servers attempt to deliver BOOTREPLY messages directly to a client using unicast delivery. The destination address in the IP header is set to the BOOTP *your IP address* and the MAC address is set to the BOOTP *client hardware address*. If a host is unable to receive a unicast IP datagram until it knows its IP address, this broadcast bit must be set to indicate to the server that the BOOTREPLY must be sent as an IP and MAC broadcast. Otherwise, this bit must be set to zero.

Client IP address Set by the client, either to its known IP address or 0.0.0.0.

Your IP address Set by the server if the client IP address field was 0.0.0.0.

Server IP address Set by the server.

Router IP address This is the address of a BOOTP relay agent, *not* a general IP router to be used by the client. It is set by the forwarding agent when BOOTP forwarding is used (see 3.6.1, “BOOTP forwarding” on page 129).

Client hardware address
Set by the client and used by the server to identify which registered client is booting.

Server host name Optional server host name terminated by X'00'.

Boot file name The client either leaves this null or specifies a generic name, such as router indicating the type of boot file to be used. The server returns the fully qualified file name of a boot file suitable for the client. The value is terminated by X'00'.

Vendor-specific area Optional vendor-specific area. Clients should always fill the first four bytes with a “magic cookie.” If a vendor-specific magic cookie is not used, the client should use 99.130.83.99 followed by an end tag (255) and set the remaining bytes to zero. The vendor-specific area can also contain *BOOTP Vendor extensions*. These are options that can be passed to the client at boot time along with its IP address. For example, the client can also receive the address of a default router, the address of a domain name server, and a subnet mask. BOOTP shares the same options as DHCP, with the exception of several DHCP-specific options. See RFC 2132 for full details.

After the BOOTP client has processed the reply, it can proceed with the transfer of the boot file and execute the full boot process. See RFC 906 for the specification of how this is done with TFTP. In the case of a diskless host, the full boot process will normally replace the minimal IP protocol stack, loaded from ROM, and used by BOOTP and TFTP, with a normal IP protocol stack transferred as part of the boot file and containing the correct customization for the client.

3.6.1 BOOTP forwarding

The BOOTP client uses the limited broadcast address for BOOTP requests, which requires the BOOTP server to be on the same subnet as the client. BOOTP forwarding is a mechanism for routers to forward BOOTP requests across subnets. It is a configuration option available on most routers. The router configured to forward BOOTP requests is known as a *BOOTP relay agent*.

A router will normally discard any datagrams containing illegal source addresses, such as 0.0.0.0, which is used by a BOOTP client. A router will also generally discard datagrams with the limited broadcast destination address. However, a BOOTP relay agent will accept such datagrams from BOOTP clients on port 67. The process carried out by a BOOTP relay agent on receiving a BOOTPREQUEST is as follows:

1. When the BOOTP relay agent receives a BOOTPREQUEST, it first checks the hops field to check the number of hops already completed in order to decide whether to forward the request. The threshold for the allowable number of hops is normally configurable.
2. If the relay agent decides to relay the request, it checks the contents of the router IP address field. If this field is zero, it fills this field with the IP address of the interface on which the BOOTPREQUEST was received. If this field already has an IP address of another relay agent, it is not touched.

3. The value of the hops field is incremented.
4. The relay agent then forwards the BOOTPREQUEST to one or more BOOTP servers. The address of the BOOTP server or servers is preconfigured at the relay agent. The BOOTPREQUEST is normally forwarded as a unicast frame, although some implementations use broadcast forwarding.
5. When the BOOTP server receives the BOOTPREQUEST with the non-zero router IP address field, it sends an IP unicast BOOTREPLY to the BOOTP relay agent at the address in this field on port 67.
6. When the BOOTP relay agent receives the BOOTREPLY, the H/W type, length, and client hardware address fields in the message supply sufficient link layer information to return the reply to the client. The relay agent checks the broadcast flag. If this flag is set, the agent forwards the BOOTPREPLY to the client as a broadcast. If the broadcast flag is not set, the relay agent sends a reply as a unicast to the address specified in your IP address.

When a router is configured as a BOOTP relay agent, the BOOTP forwarding task is considerably different from the task of switching datagrams between subnets normally carried out by a router. Forwarding of BOOTP messages can be considered to be receiving BOOTP messages as a final destination, and then generating new BOOTP messages to be forwarded to another destination.

3.6.2 BOOTP considerations

The use of BOOTP allows centralized configuration of multiple clients. However, it requires a static table to be maintained with an IP address preallocated for every client that is likely to attach to the BOOTP server, even if the client is seldom active. This means that there is no relief on the number of IP addresses required. There is a measure of security in an environment using BOOTP, because a client will only be allocated an IP address by the server if it has a valid MAC address.

3.7 Dynamic Host Configuration Protocol (DHCP)

DHCP is a draft standard protocol. Its status is elective. The current DHCP specifications are in RFC 2131 with updates in RFC 3396 and RFC 4361. The specifications are also in RFC 2132 with updates in RFC3442, RFC3942, and RFC4361.

The Dynamic Host Configuration Protocol (DHCP) provides a framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on the BOOTP protocol, adding the capability of automatic allocation of reusable network addresses and additional configuration options. For information

regarding BOOTP, refer to 3.6, “Bootstrap Protocol (BOOTP)” on page 125. DHCP messages use UDP port 67, the BOOTP server's well-known port and UDP port 68, the BOOTP client's well-known port. DHCP participants can interoperate with BOOTP participants. See 3.7.8, “BOOTP and DHCP interoperability” on page 140 for further details.

DHCP consists of two components:

- A protocol that delivers host-specific configuration parameters from a DHCP server to a host
- A mechanism for the allocation of temporary or permanent network addresses to hosts

IP requires the setting of many parameters within the protocol implementation software. Because IP can be used on many dissimilar kinds of network hardware, values for those parameters cannot be guessed at or assumed to have correct defaults. The use of a distributed address allocation scheme based on a polling/defense mechanism, for discovery of network addresses already in use, cannot guarantee unique network addresses because hosts might not always be able to defend their network addresses.

DHCP supports three mechanisms for IP address allocation:

- Automatic allocation

  DHCP assigns a permanent IP address to the host.
- Dynamic allocation

  DHCP assigns an IP address for a limited period of time. Such a network address is called a *lease*. This is the only mechanism that allows automatic reuse of addresses that are no longer needed by the host to which it was assigned.
- Manual allocation

  The host's address is assigned by a network administrator.

Archived

3.7.1 The DHCP message format

The format of a DHCP message is shown in Figure 3-44.

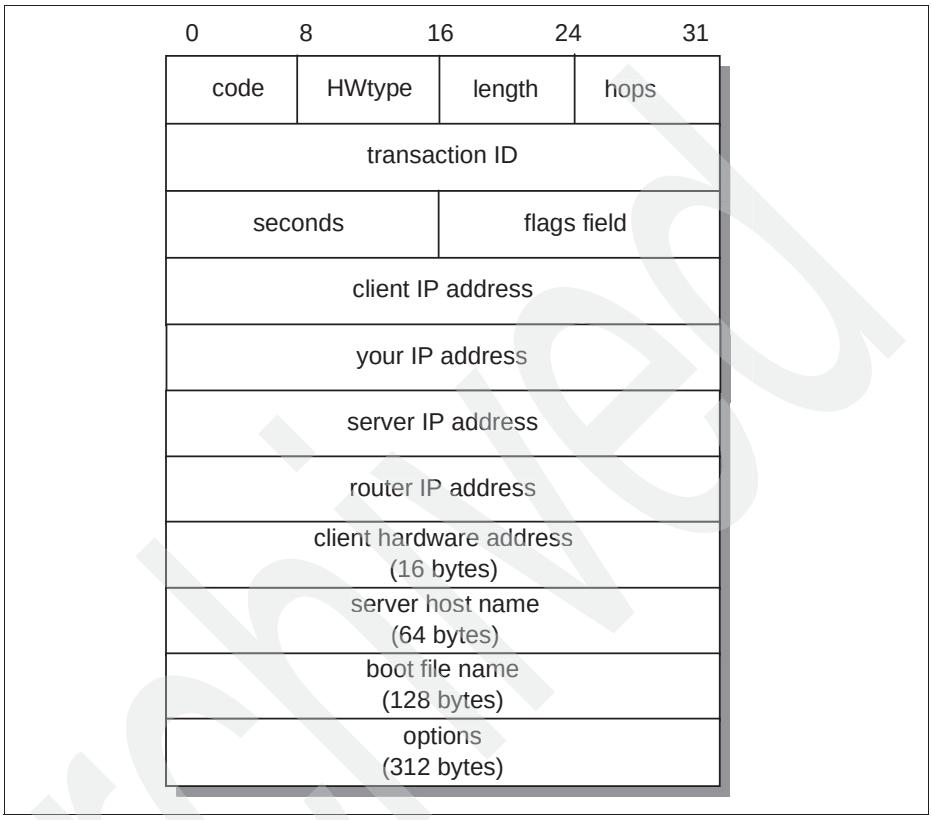


Figure 3-44 DHCP message format

Where:

Code Indicates a request or a reply:

1 Request
2 Reply

HWtype The type of hardware, for example:

1 Ethernet
6 IEEE 802 Networks

Refer to STD 2 – Assigned Internet Numbers for a complete list.

Length Hardware address length in bytes.

Archived

Hops	The client sets this to 0. It is incremented by a router that relays the request to another server and is used to identify loops. RFC 951 suggests that a value of 3 indicates a loop.
Transaction ID	A random number used to match this boot request with the response it generates.
Seconds	Set by the client. It is the elapsed time in seconds since the client started its boot process.
Flags field	The most significant bit of the flags field is used as a broadcast flag. All other bits must be set to zero, and are reserved for future use. Normally, DHCP servers attempt to deliver DHCP messages directly to a client using unicast delivery. The destination address in the IP header is set to the DHCP *your IP address* and the MAC address is set to the DHCP *client hardware address*. If a host is unable to receive a unicast IP datagram until it knows its IP address, this broadcast bit must be set to indicate to the server that the DHCP reply must be sent as an IP and MAC broadcast. Otherwise, this bit must be set to zero.
Client IP address	Set by the client. Either its known IP address, or 0.0.0.0.
Your IP address	Set by the server if the client IP address field was 0.0.0.0.
Server IP address	Set by the server.
Router IP address	This is the address of a BOOTP relay agent, *not* a general IP router to be used by the client. It is set by the forwarding agent when BOOTP forwarding is used (see 3.6.1, “BOOTP forwarding” on page 129).
Client hardware address	Set by the client. DHCP defines a client identifier option that is used for client identification. If this option is not used, the client is identified by its MAC address.
Server host name	Optional server host name terminated by X'00'.
Boot file name	The client either leaves this null or specifies a generic name, such as router, indicating the type of boot file to be used. In a DHCPDISCOVER request, this is set to null. The server returns a fully qualified directory path name in a DHCPOFFER request. The value is terminated by X'00'.

Archive

Options The first four bytes of the options field of the DHCP message contain the magic cookie (99.130.83.99). The remainder of the options field consists of tagged parameters that are called *options*. See RFC 2132, with updates in RFC3942, for details.

3.7.2 DHCP message types

DHCP messages fall into one of the following categories:

- DHCPDISCOVER: Broadcast by a client to find available DHCP servers.
- DHCPOFFER: Response from a server to a DHCPDISCOVER and offering IP address and other parameters.
- DHCPREQUEST: Message from a client to servers that does one of the following:
  - Requests the parameters offered by one of the servers and declines all other offers.
  - Verifies a previously allocated address after a system or network change (a reboot for example).
  - Requests the extension of a lease on a particular address.
- DHCPACK: Acknowledgement from server to client with parameters, including IP address.
- DHCPNACK: Negative acknowledgement from server to client, indicating that the client's lease has expired or that a requested IP address is incorrect.
- DHCPDECLINE: Message from client to server indicating that the offered address is already in use.
- DHCPRELEASE: Message from client to server cancelling remainder of a lease and relinquishing network address.
- DHCPINFORM: Message from a client that already has an IP address (manually configured, for example), requesting further configuration parameters from the DHCP server.

3.7.3 Allocating a new network address

This section describes the client/server interaction if the client does not know its network address. Assume that the DHCP server has a block of network addresses from which it can satisfy requests for new addresses. Each server also maintains a database of allocated addresses and leases in permanent local storage.

The DHCP client/server interaction steps are illustrated in Figure 3-45.

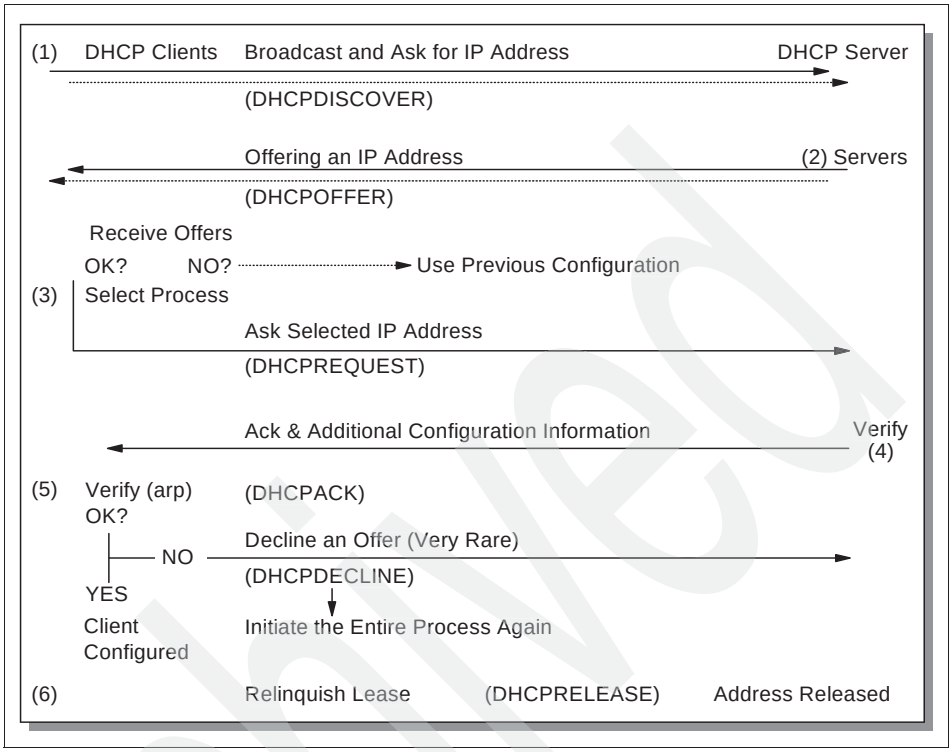


Figure 3-45 DHCP client and DHCP server interaction

The following procedure describes the DHCP client/server interaction steps illustrated in Figure 3-45:

1. The client broadcasts a DHCPDISCOVER message on its local physical subnet. At this point, the client is in the INIT state. The DHCPDISCOVER message might include some options such as network address suggestion or lease duration.
2. Each server responds with a DHCPOFFER message that includes an available network address (your IP address) and other configuration options. The servers record the address as offered to the client to prevent the same address being offered to other clients in the event of further DHCPDISCOVER messages being received before the first client has completed its configuration.

Archived

3. The client receives one or more DHCPOFFER messages from one or more servers. The client chooses one based on the configuration parameters offered and broadcasts a DHCPREQUEST message that includes the server identifier option to indicate which message it has selected and the requested IP address option taken from your IP address in the selected offer.
4. In the event that no offers are received, if the client has knowledge of a previous network address, the client can reuse that address if its lease is still valid until the lease expires.
5. The servers receive the DHCPREQUEST broadcast from the client. Those servers not selected by the DHCPREQUEST message use the message as notification that the client has declined that server's offer. The server selected in the DHCPREQUEST message commits the binding for the client to persistent storage and responds with a DHCPACK message containing the configuration parameters for the requesting client. The combination of client hardware and assigned network address constitute a unique identifier for the client's lease and are used by both the client and server to identify a lease referred to in any DHCP messages. The your IP address field in the DHCPACK messages is filled in with the selected network address.
6. The client receives the DHCPACK message with configuration parameters. The client performs a final check on the parameters, for example, with ARP for allocated network address, and notes the duration of the lease and the lease identification cookie specified in the DHCPACK message. At this point, the client is configured.
7. If the client detects a problem with the parameters in the DHCPACK message (the address is already in use in the network, for example), the client sends a DHCPDECLINE message to the server and restarts the configuration process. The client should wait a minimum of ten seconds before restarting the configuration process to avoid excessive network traffic in case of looping. On receipt of a DHCPDECLINE, the server must mark the offered address as unavailable (and possibly inform the system administrator that there is a configuration problem).
8. If the client receives a DHCPNAK message, the client restarts the configuration process.
9. The client may choose to relinquish its lease on a network address by sending a DHCPRELEASE message to the server. The client identifies the lease to be released by including its network address and its hardware address.

3.7.4 DHCP lease renewal process

This section describes the interaction between DHCP servers and clients that have already been configured and the process that ensures lease expiration and renewal.

The process involves the following steps:

1. When a server sends the DHCPACK to a client with IP address and configuration parameters, it also registers the start of the lease time for that address. This lease time is passed to the client as one of the options in the DHCPACK message, together with two timer values, T1 and T2. The client is rightfully entitled to use the given address for the duration of the lease time. On applying the received configuration, the client also starts the timers T1 and T2. At this time, the client is in the BOUND state. Times T1 and T2 are options configurable by the server, but T1 must be less than T2, and T2 must be less than the lease time. According to RFC 2132, T1 defaults to (0.5 * lease time) and T2 defaults to (0.875 * lease time).
2. When timer T1 expires, the client will send a DHCPREQUEST (unicast) to the server that offered the address, asking to extend the lease for the given configuration. The client is now in the RENEWING state. The server usually responds with a DHCPACK message indicating the new lease time, and timers T1 and T2 are reset at the client accordingly. The server also resets its record of the lease time. In normal circumstances, an active client continually renews its lease in this way indefinitely, without the lease ever expiring.
3. If no DHCPACK is received until timer T2 expires, the client enters the REBINDING state. It now broadcasts a DHCPREQUEST message to extend its lease. This request can be confirmed by a DHCPACK message from any DHCP server in the network.
4. If the client does not receive a DHCPACK message after its lease has expired, it has to stop using its current TCP/IP configuration. The client can then return to the INIT state, issuing a DHCPDISCOVER broadcast to try and obtain any valid address.

Archive

Figure 3-46 shows the DHCP process and changing client state during that process.

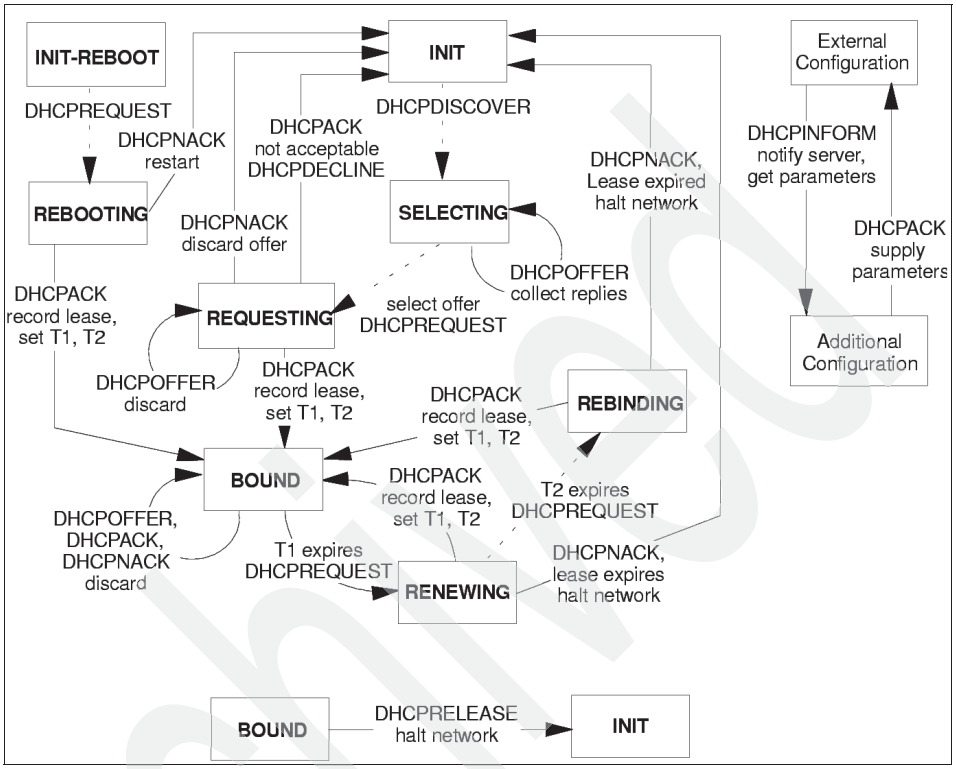


Figure 3-46 DHCP client state and DHCP process

3.7.5 Reusing a previously allocated network address

If the client remembers and wants to reuse a previously allocated network address, the following steps are carried out:

1. The client broadcasts a DHCPREQUEST message on its local subnet. The DHCPREQUEST message includes the client's network address.
2. A server with knowledge of the client's configuration parameters responds with a DHCPACK message to the client (provided the lease is still current), renewing the lease at the same time.
3. If the client's lease has expired, the server with knowledge of the client responds with DHCPNACK.

4. The client receives the DHCPACK message with configuration parameters. The client performs a final check on the parameters and notes the duration of the lease and the lease identification cookie specified in the DHCPACK message. At this point, the client is configured and its T1 and T2 timers are reset.
5. If the client detects a problem with the parameters in the DHCPACK message, the client sends a DHCPDECLINE message to the server and restarts the configuration process by requesting a new network address. If the client receives a DHCPNAK message, it cannot reuse its remembered network address. It must instead request a new address by restarting the configuration process as described in 3.7.3, “Allocating a new network address” on page 134.

For further information, refer to the previously mentioned RFCs.

3.7.6 Configuration parameters repository

DHCP provides persistent storage of network parameters for network clients. A DHCP server stores a key-value entry for each client, the key being some unique identifier, for example, an IP subnet number and a unique identifier within the subnet (normally a hardware address), and the value contains the configuration parameters last allocated to this particular client.

One effect of this is that a DHCP client will tend always to be allocated to the same IP address by the server, provided the pool of addresses is not over-subscribed and the previous address has not already been allocated to another client.

3.7.7 DHCP considerations

DHCP dynamic allocation of IP addresses and configuration parameters relieves the network administrator of a great deal of manual configuration work. The ability for a device to be moved from network to network and to automatically obtain valid configuration parameters for the current network can be of great benefit to mobile users. Also, because IP addresses are only allocated when clients are actually active, it is possible, by the use of reasonably short lease times and the fact that mobile clients do not need to be allocated more than one address, to reduce the total number of addresses in use in an organization. However, consider the following points when DHCP is implemented:

- DHCP is built on UDP, which is inherently insecure. In normal operation, an unauthorized client can connect to a network and obtain a valid IP address and configuration. To prevent this, it is possible to preallocate IP addresses to particular MAC addresses (similar to BOOTP), but this increases the

administration workload and removes the benefit of recycling of addresses. Unauthorized DHCP servers can also be set up, sending false and potentially disruptive information to clients.

- In a DHCP environment where automatic or dynamic address allocation is used, it is generally not possible to predetermine the IP address of a client at any particular point in time. In this case, if static DNS servers are also used, the DNS servers will not likely contain valid host name to IP address mappings for the clients. If having client entries in the DNS is important for the network, you can use DHCP to manually assign IP addresses to those clients and then administer the client mappings in the DNS accordingly.

3.7.8 BOOTP and DHCP interoperability

The format of DHCP messages is based on the format of BOOTP messages, which enables BOOTP and DHCP clients to interoperate in certain circumstances. Every DHCP message contains a DHCP message type (51) option. Any message without this option is assumed to be from a BOOTP client.

Support for BOOTP clients at a DHCP server must be configured by a system administrator, if required. The DHCP server responds to BOOTPREQUEST messages with BOOTPREPLY, rather than DHCPOFFER. Any DHCP server that is not configured in this way will discard any BOOTPREQUEST frames sent to it. A DHCP server can offer static addresses, or automatic addresses (from its pool of unassigned addresses), to a BOOTP client (although not all BOOTP implementations will understand automatic addresses). If an automatic address *is* offered to a BOOTP client, that address must have an infinite lease time, because the client will not understand the DHCP lease mechanism.

DHCP messages can be forwarded by routers configured as BOOTP relay agents.

3.8 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 791 – Internet Protocol (September 1981)
- RFC 792 – Internet Control Message Protocol (September 1981)
- RFC 826 – Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware (November 1982)
- RFC 903 – A Reverse Address Resolution Protocol (June 1984)

- RFC 906 – Bootstrap loading using TFTP (June 1984)
- RFC919 – Broadcasting Internet Datagrams (October 1984)
- RFC922 – Broadcasting Internet datagrams in the presence of subnets (October 1984)
- RFC 925 – Multi-LAN address resolution (October 1984)
- RFC 950 – Internet Standard Subnetting Procedure (August 1985)
- RFC 951 – Bootstrap Protocol (September 1985)
- RFC 1027 – Using ARP to implement transparent subnet gateways (October 1987)
- RFC 1112 – Host extensions for IP multicasting (August 1989)
- RFC 1122 – Requirements for Internet Hosts – Communication Layers (October 1989)
- RFC 1166 – Internet numbers (July 1990)
- RFC 1191 – Path MTU discovery (November 1990)
- RFC 1256 – ICMP Router Discovery Messages (September 1991)
- RFC 1349 – Type of Service in the Internet Protocol Suite (July 1992)
- RFC 1393 Traceroute Using an IP Option G (January 1993)
- RFC 1466 – Guidelines for Management of IP Address Space (May 1993)
- RFC 1518 – An Architecture for IP Address Allocation with CIDR (September 1993)
- RFC 1519 – Classless Inter-Domain Routing (CIDR): an Address Assignment (September 1993)
- RFC 1520 – Exchanging Routing Information Across Provider Boundaries in the CIDR Environment (September 1993)
- RFC 1542 – Clarifications and Extensions for the Bootstrap Protocol (October 1993)
- RFC 1788 – ICMP Domain Name Messages (April 1995)
- RFC 1812 – Requirements for IP Version 4 Routers (June 1995)
- RFC 1918 – Address Allocation for Private Internets (February 1996)
- RFC 2050 – Internet Registry IP Allocation Guidelines (November 1996)
- RFC 2131 – Dynamic Host Configuration Protocol (March 1997)
- RFC 2132 – DHCP Options and BOOTP Vendor Extensions (March 1997)
- RFC 2236 – Internet Group Management Protocol, Version 2 (November 1997)

Archived

- RFC 2474 – Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (December 1998)
- RFC 2644 – Changing the Default for Directed Broadcasts in Router (August 1999)
- RFC 2663 – IP Network Address Translator (NAT) Terminology and Considerations (August 1999)
- RFC 3022 – Traditional IP Network Address Translator (Traditional NAT) (January 2001)
- RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP (September 2001)
- RFC 3260 – New Terminology and Clarifications for Diffserv (April 2002)
- RFC 3330 – Special-Use IPv4 Addresses (September 2002)
- RFC 3396 – Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4) (November 2002)
- RFC 3442 – The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) version 4 (December 2002)
- RFC 3942 – Reclassifying Dynamic Host Configuration Protocol version 4 (DHCPv4) Options (November 2004)
- RFC 4361 – Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4) (February 2006)
- RFC 4379 – Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures (February 2006)

Archived

4

Transport layer protocols

This chapter provides an overview of the most important and commonly used protocols of the TCP/IP transport layer. These include:

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

By building on the functionality provided by the Internet Protocol (IP), the transport protocols deliver data to applications executing in the internet. This is done by making use of ports, as described in 4.1, “Ports and sockets” on page 144. The transport protocols can provide additional functionality such as congestion control, reliable data delivery, duplicate data suppression, and flow control as is done by TCP.

© Copyright IBM Corp. 1989-2006. All rights reserved.

4.1 Ports and sockets

This section introduces the concepts of the port and socket, which are needed to determine which local process at a given host actually communicates with which process, at which remote host, using which protocol. If this sounds confusing, consider the following points:

- An application process is assigned a process identifier number (process ID), which is likely to be different each time that process is started.
- Process IDs differ between operating system platforms, thus they are not uniform.
- A server process can have multiple connections to multiple clients at a time, thus simple connection identifiers are not unique.The concept of ports and sockets provides a way to uniformly and uniquely identify connections and the programs and hosts that are engaged in them, irrespective of specific process IDs.

The concept of ports and sockets provides a way to uniformly and uniquely identify connections and the programs and hosts that are engaged in them, irrespective of specific process IDs.

4.1.1 Ports

Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number used by the host-to-host protocol to identify to which higher-level protocol or application program (process) it must deliver incoming messages. There are two types of ports:

- Well-known: Well-known ports belong to standard servers, for example, Telnet uses port 23. Well-known port numbers range between 1 and 1023 (prior to 1992, the range between 256 and 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port. Exceptions are the BOOTP server, which uses two: 67 and 68 (see 3.6, “Bootstrap Protocol (BOOTP)” on page 125) and the FTP server, which uses two: 20 and 21 (see (14.1, “File Transfer Protocol (FTP)” on page 514).

  The well-known ports are controlled and assigned by the Internet Assigned Number Authority (IANA) and on most systems can only be used by system processes or by programs executed by privileged users. Well-known ports allow clients to find servers without configuration information. The well-known port numbers are defined in STD 2 – Assigned Internet Numbers.

- Ephemeral: Some clients do not need well-known port numbers because they initiate communication with servers, and the port number they are using is contained in the UDP/TCP datagrams sent to the server. Each client process is allocated a port number, for as long as it needs, by the host on which it is running. Ephemeral port numbers have values greater than 1023, normally in the range of 1024 to 65535.

  Ephemeral ports are not controlled by IANA and can be used by ordinary user-developed programs on most systems.

Confusion, due to two different applications trying to use the same port numbers on one host, is avoided by writing those applications to request an available port from TCP/IP. Because this port number is dynamically assigned, it can differ from one invocation of an application to the next.

UDP, TCP, and ISO TP-4 all use the same port principle. To the best possible extent, the same port numbers are used for the same services on top of UDP, TCP, and ISO TP-4.

Note: Normally, a server will use either TCP or UDP, but there are exceptions. For example, domain name servers (see 12.1, “Domain Name System (DNS)” on page 426) use both UDP port 53 and TCP port 53.

4.1.2 Sockets

The socket interface is one of several application programming interfaces to the communication protocols (see 11.2, “Application programming interfaces (APIs)” on page 410). Designed to be a generic communication programming interface, socket APIs were first introduced by 4.2 Berkeley Software Distribution (BSD). Although it has not been standardized, Berkeley socket API has become a de facto industry standard abstraction for network TCP/IP socket implementation.

Consider the following terminologies:

- A *socket* is a special type of *file handle*, which is used by a process to request network services from the operating system.
- A *socket address* is the triple:

  <protocol, local-address, local port>

  For example, in the TCP/IP (version 4) suite:

  <tcp, 192.168.14.234, 8080>
- A *conversation* is the communication link between two processes.

- An *association* is the 5-tuple that completely specifies the two processes that comprise a connection:

  <protocol, local-address, local-port, foreign-address, foreign-port>

  In the TCP/IP (version 4) suite, the following could be a valid association:

  <tcp, 192.168.14.234, 1500, 192.168.44, 22>

- A *half-association* is either one of the following, which each specify half of a connection:

  <protocol, local-address, local-process>

  Or:

  <protocol, foreign-address, foreign-process>

  The half-association is also called a *socket* or a *transport address*. That is, a socket is an endpoint for communication that can be named and addressed in a network.

Two processes communicate through TCP sockets. The socket model provides a process with a full-duplex byte stream connection to another process. The application need not concern itself with the management of this stream; these facilities are provided by TCP.

TCP uses the same port principle as UDP to provide multiplexing. Like UDP, TCP uses well-known and ephemeral ports. Each side of a TCP connection has a socket that can be identified by the triple <TCP, IP address, port number>. If two processes are communicating over TCP, they have a logical connection that is uniquely identifiable by the two sockets involved, that is, by the combination <TCP, local IP address, local port, remote IP address, remote port>. Server processes are able to manage multiple conversations through a single port. Refer to 11.2.1, “The socket API” on page 410 for more information about socket APIs.

4.2 User Datagram Protocol (UDP)

UDP is a standard protocol with STD number 6. UDP is described by RFC 768 – User Datagram Protocol. Its status is standard and almost every TCP/IP implementation intended for small data units transfer or those which can afford to lose a little amount of data (such as multimedia streaming) will include UDP.

UDP is basically an application interface to IP. It adds no reliability, flow-control, or error recovery to IP. It simply serves as a multiplexer/demultiplexer for sending and receiving datagrams, using ports to direct the datagrams, as shown in Figure 4-1. For a more detailed discussion of ports, refer to 4.1, “Ports and sockets” on page 144.

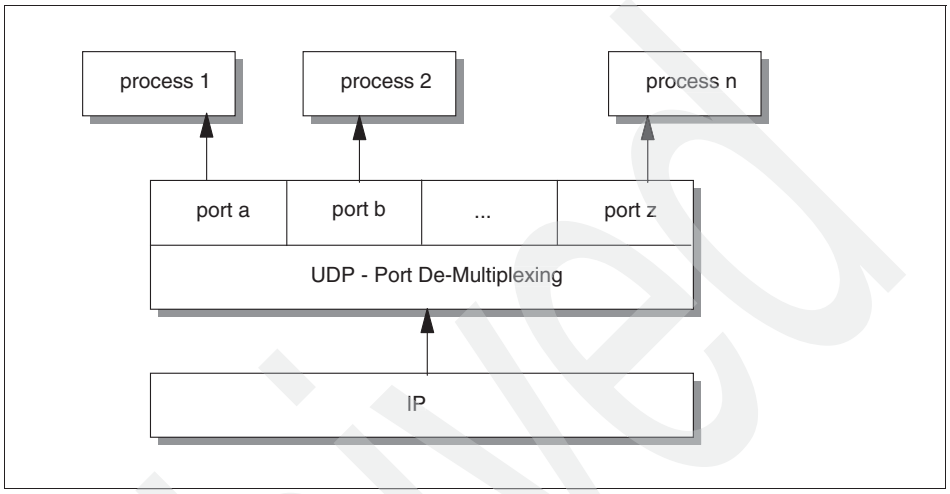


Figure 4-1 UDP: Demultiplexing based on ports

UDP provides a mechanism for one application to send a datagram to another. The UDP layer can be regarded as being extremely thin and is, consequently, very efficient, but it requires the application to take responsibility for error recovery and so on.

Applications sending datagrams to a host need to identify a target that is more specific than the IP address, because datagrams are normally directed to certain processes and not to the system as a whole. UDP provides this by using ports. We discuss the port concept in 4.1, “Ports and sockets” on page 144.

4.2.1 UDP datagram format

Each UDP datagram is sent within a single IP datagram. Although, the IP datagram might be fragmented during transmission, the receiving IP implementation will reassemble it before presenting it to the UDP layer. All IP implementations are required to accept datagrams of 576 bytes, which means that, allowing for maximum-size IP header of 60 bytes, a UDP datagram of 516 bytes is acceptable to all implementations. Many implementations will accept larger datagrams, but this is not guaranteed.

The UDP datagram has an 8-byte header, as described in Figure 4-2 on page 148.

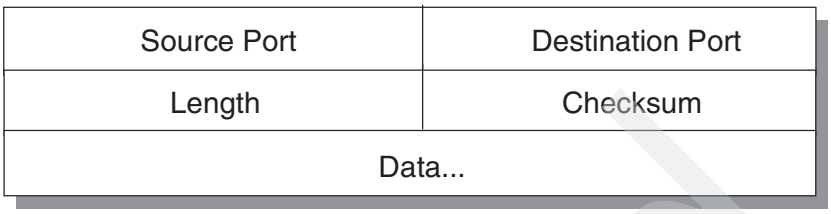


Figure 4-2 UDP: Datagram format

Where:

Source Port Indicates the port of the sending process. It is the port to which replies are addressed.

Destination Port Specifies the port of the destination process on the destination host.

Length The length (in bytes) of this user datagram, including the header.

Checksum An optional 16-bit one's complement of the one's complement sum of a pseudo-IP header, the UDP header, and the UDP data. In Figure 4-3, we see a pseudo-IP header. It contains the source and destination IP addresses, the protocol, and the UDP length.

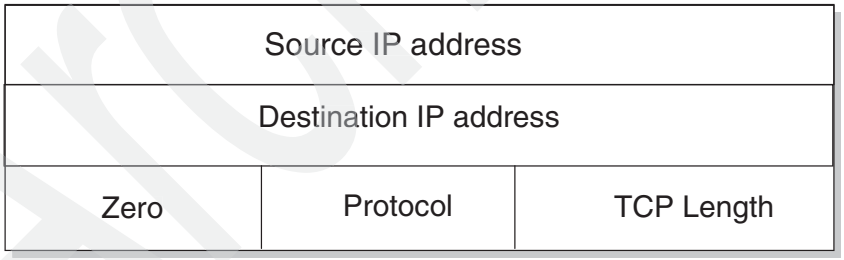


Figure 4-3 UDP: Pseudo-IP header

The pseudo-IP header effectively extends the checksum to include the original (unfragmented) IP datagram.

4.2.2 UDP application programming interface

The application interface offered by UDP is described in RFC 768. It provides for:

- The creation of new receive ports
- The receive operation that returns the data bytes and an indication of source port and source IP address
- The send operation that has, as parameters, the data, source, and destination ports and addresses

The way this interface is implemented is left to the discretion of each vendor.

Be aware that UDP and IP do not provide guaranteed delivery, flow-control, or error recovery, so these must be provided by the application.

Standard applications using UDP include:

- Trivial File Transfer Protocol (see 14.2, “Trivial File Transfer Protocol (TFTP)” on page 529).
- Domain Name System name server (see 12.2, “Dynamic Domain Name System” on page 453).
- Remote Procedure Call, used by the Network File System (see both 11.2.2, “Remote Procedure Call (RPC)” on page 415 and 14.4, “Network File System (NFS)” on page 538).
- Simple Network Management Protocol (see 17.1, “The Simple Network Management Protocol (SNMP)” on page 624).
- Lightweight Directory Access Protocol (see 12.4, “Lightweight Directory Access Protocol (LDAP)” on page 459).

4.3 Transmission Control Protocol (TCP)

TCP is a standard protocol with STD number 7. TCP is described by RFC 793 – Transmission Control Protocol. Its status is standard, and in practice, every TCP/IP implementation that is not used exclusively for routing will include TCP.

TCP provides considerably more facilities for applications than UDP. Specifically, this includes error recovery, flow control, and reliability. TCP is a *connection-oriented* protocol, unlike UDP, which is *connectionless*. Most of the user application protocols, such as Telnet and FTP, use TCP. The two processes communicate with each other over a TCP connection (InterProcess

Archived

Communication, or IPC), as shown in Figure 4-4. In the figure, processes 1 and 2 communicate over a TCP connection carried by IP datagrams. See 4.1, “Ports and sockets” on page 144 for more details about ports and sockets.

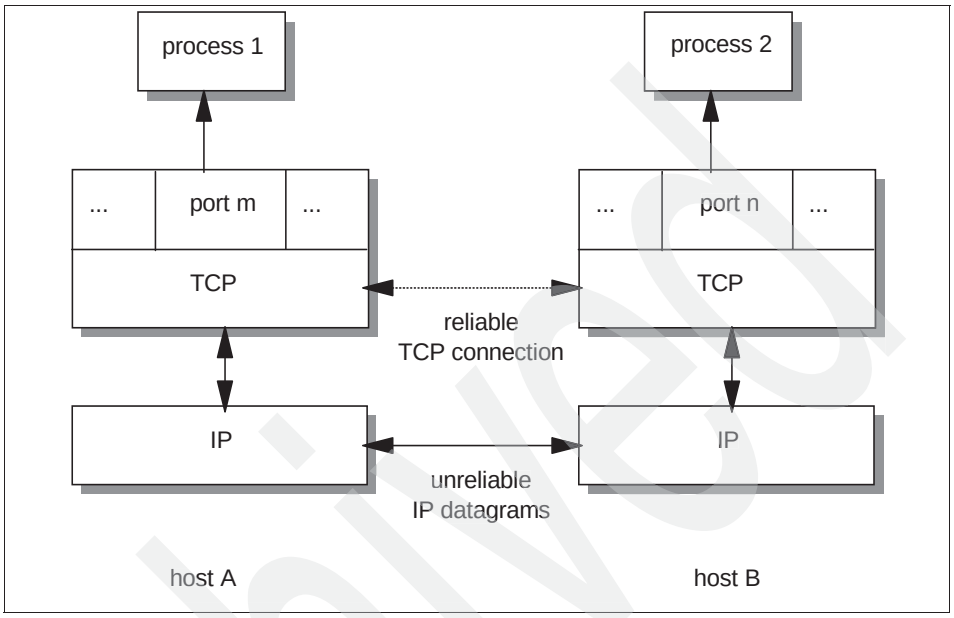


Figure 4-4 TCP: Connection between processes

4.3.1 TCP concept

As noted earlier, the primary purpose of TCP is to provide a reliable logical circuit or connection service between pairs of processes. It does *not* assume reliability from the lower-level protocols (such as IP), so TCP must guarantee this itself.

TCP can be characterized by the following facilities it provides for the applications using it:

- Stream data transfer: From the application's viewpoint, TCP transfers a contiguous stream of bytes through the network. The application does not have to bother with chopping the data into basic blocks or datagrams. TCP does this by grouping the bytes into TCP segments, which are passed to the IP layer for transmission to the destination. Also, TCP itself decides how to segment the data, and it can forward the data at its own convenience.

  Sometimes, an application needs to be sure that all the data passed to TCP has actually been transmitted to the destination. For that reason, a push function is defined. It will push all remaining TCP segments still in storage to

the destination host. The normal close connection function also pushes the data to the destination.

- Reliability: TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP layer. If the ACK is not received within a timeout interval, the data is retransmitted. Because the data is transmitted in blocks (TCP segments), only the sequence number of the first data byte in the segment is sent to the destination host.

  The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.

- Flow control: The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive (beyond the last received TCP segment) without causing overrun and overflow in its internal buffers. This is sent in the ACK in the form of the highest sequence number it can receive without problems. This mechanism is also referred to as a window-mechanism, and we discuss it in more detail later in this chapter.

- Multiplexing: Achieved through the use of ports, just as with UDP.

- Logical connections: The reliability and flow control mechanisms described here require that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.

- Full duplex: TCP provides for concurrent data streams in both directions.

Archive

The window principle

A simple transport protocol might use the following principle: send a packet and then wait for an acknowledgment from the receiver before sending the next packet. If the ACK is not received within a certain amount of time, retransmit the packet. See Figure 4-5 for more details.

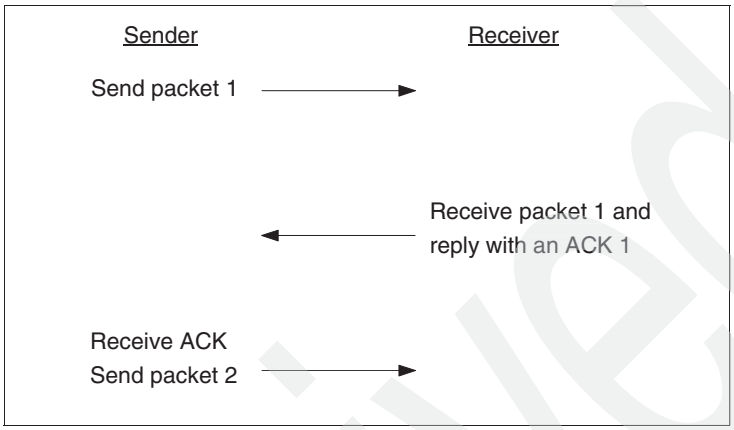


Figure 4-5 TCP: The window principle

Although this mechanism ensures reliability, it only uses a part of the available network bandwidth.

Now, consider a protocol where the sender groups its packets to be transmitted, as in Figure 4-6, and uses the following rules:

- The sender can send all packets within the window without receiving an ACK, but must start a timeout timer for each of them.
- The receiver must acknowledge each packet received, indicating the sequence number of the last well-received packet.
- The sender slides the window on each ACK received.

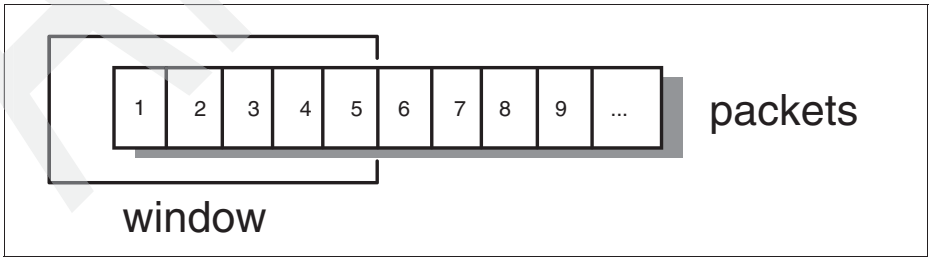


Figure 4-6 TCP: Message packets

As shown in Figure 4-7, the sender can transmit packets 1 to 5 without waiting for any acknowledgment.

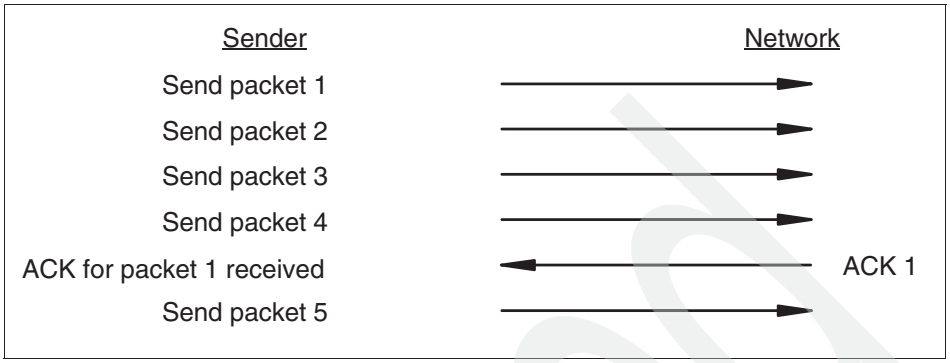


Figure 4-7 TCP: Window principle

As shown in Figure 4-8, at the moment the sender receives ACK 1 (acknowledgment for packet 1), it can slide its window one packet to the right.

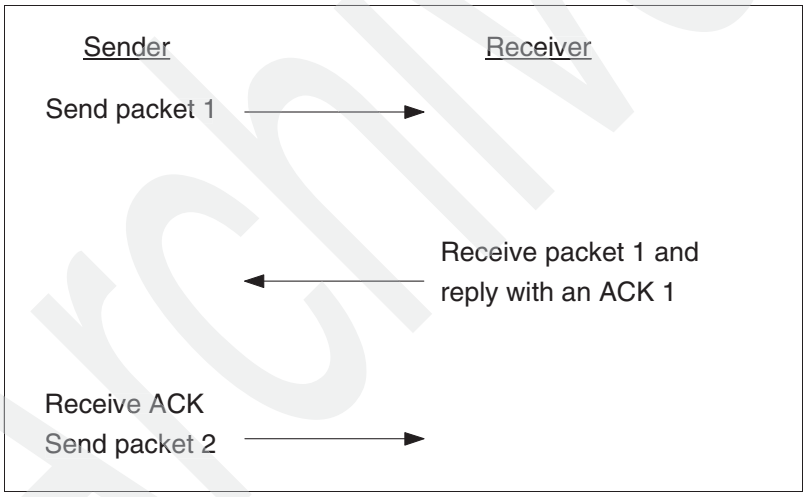


Figure 4-8 TCP: Message packets

At this point, the sender can also transmit packet 6.

Archived

Imagine some special cases:

- Packet 2 gets lost: The sender will not receive ACK 2, so its window will remain in position 1 (as in Figure 4-8 on page 153). In fact, because the receiver did not receive packet 2, it will acknowledge packets 3, 4, and 5 with an ACK 1, because packet 1 was the last one received in sequence. At the sender's side, eventually a timeout will occur for packet 2 and it will be retransmitted. Note that reception of this packet by the receiver will generate ACK 5, because it has now successfully received all packets 1 to 5, and the sender's window will slide four positions upon receiving this ACK 5.
- Packet 2 did arrive, but the acknowledgment gets lost: The sender does not receive ACK 2, but will receive ACK 3. ACK 3 is an acknowledgment for *all* packets up to 3 (including packet 2) and the sender can now slide its window to packet 4.

This window mechanism ensures:

- Reliable transmission.
- Better use of the network bandwidth (better throughput).
- Flow-control, because the receiver can delay replying to a packet with an acknowledgment, knowing its free buffers are available and the window size of the communication.

The window principle applied to TCP

The previously discussed window principle is used in TCP, but with a few differences:

- Because TCP provides a byte-stream connection, sequence numbers are assigned to each byte in the stream. TCP divides this contiguous byte stream into TCP segments to transmit them. The window principle is used at the byte level, that is, the segments sent and ACKs received will carry byte-sequence numbers and the window size is expressed as a number of bytes, rather than a number of packets.
- The window size is determined by the receiver when the connection is established and is variable during the data transfer. Each ACK message will include the window size that the receiver is ready to deal with at that particular time.

Archived

The sender's data stream can now be seen as follows in Figure 4-9.

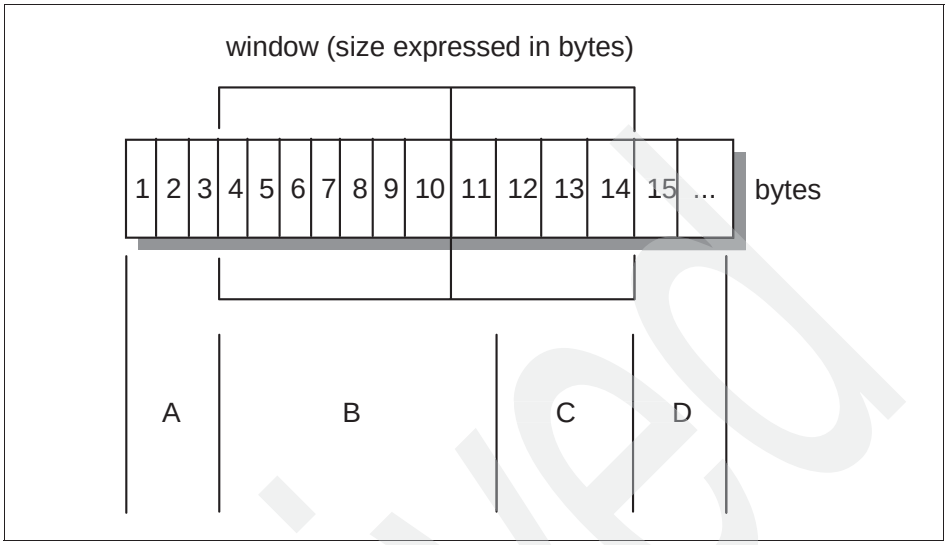


Figure 4-9 TCP: Window principle applied to TCP

Where:

A Bytes that are transmitted and have been acknowledged

B Bytes that are sent but not yet acknowledged

C Bytes that can be sent without waiting for any acknowledgment

D Bytes that cannot be sent yet

Remember that TCP will block bytes into segments, and a TCP segment only carries the sequence number of the first byte in the segment.

Archived

TCP segment format

Figure 4-10 shows the TCP segment format.

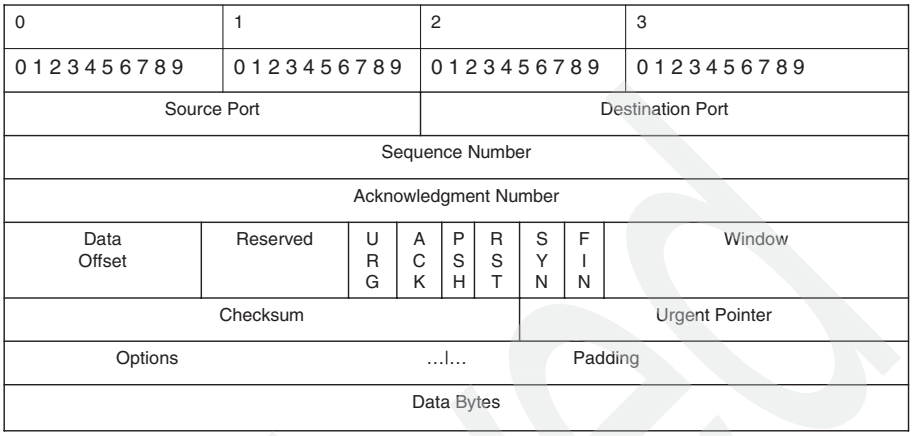


Figure 4-10 TCP: Segment format

Where:

Source Port The 16-bit source port number, used by the receiver to reply.

Destination Port The 16-bit destination port number.

Sequence Number The sequence number of the first data byte in this segment. If the SYN control bit is set, the sequence number is the initial sequence number (n) and the first data byte is n+1.

Acknowledgment Number
If the ACK control bit is set, this field contains the value of the next sequence number that the receiver is expecting to receive.

Data Offset The number of 32-bit words in the TCP header. It indicates where the data begins.

Reserved Six bits reserved for future use; must be zero.

URG Indicates that the urgent pointer field is significant in this segment.

ACK Indicates that the acknowledgment field is significant in this segment.

PSH Push function.

RST Resets the connection.

SYN Synchronizes the sequence numbers.

FIN No more data from sender.

Window Used in ACK segments. It specifies the number of data bytes, beginning with the one indicated in the acknowledgment number field that the receiver (the sender of this segment) is willing to accept.

Checksum The 16-bit one's complement of the one's complement sum of all 16-bit words in a pseudo-header, the TCP header, and the TCP data. While computing the checksum, the checksum field itself is considered zero.

The pseudo-header is the same as that used by UDP for calculating the checksum. It is a pseudo-IP-header, only used for the checksum calculation, with the format shown in Figure 4-11.

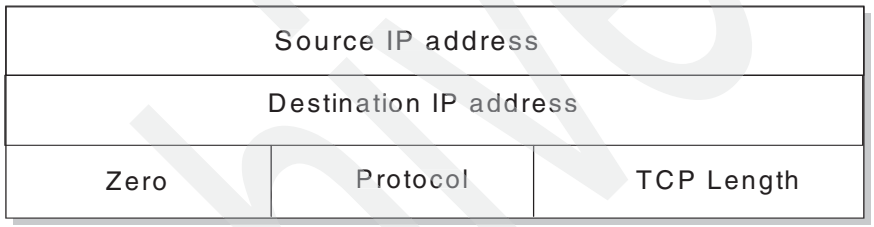


Figure 4-11 TCP: Pseudo-IP header

Urgent Pointer Points to the first data octet following the urgent data. Only significant when the URG control bit is set.

Options Just as in the case of IP datagram options, options can be either:

- A single byte containing the option number
- A variable length option in the following format as shown in Figure 4-12



Figure 4-12 TCP: IP datagram option, variable length option

Archive

There are currently seven options defined, as shown in Table 4-1.

Table 4-1 TCP: IP datagram options

Kind	Length	Meaning
0	-	End of option list
1	-	No operation
2	4	Maximum segment size
3	3	Window scale
4	2	Sack-permitted
5	X	Sack
8	10	Time stamps

Maximum segment size option

This option is only used during the establishment of the connection (SYN control bit set) and is sent from the side that is to receive data to indicate the maximum segment length it can handle. If this option is not used, any segment size is allowed. See Figure 4-13 for more details.



Figure 4-13 TCP: Maximum segment size

Archived

Window scale option

This option is not mandatory. Both sides must send the Window scale option in their SYN segments to enable windows scaling in their direction. The Window scale expands the definition of the TCP window to 32 bits. It defines the 32-bit window size by using scale factor in the SYN segment over standard 16-bit window size. The receiver rebuilds the 32-bit window size by using the 16-bit window size and scale factor. This option is determined while handshaking. There is no way to change it after the connection has been established. See Figure 4-14 for more details.



Figure 4-14 TCP: Window scale option

SACK-permitted option

This option is set when selective acknowledgment is used in that TCP connection. See Figure 4-15 for details.

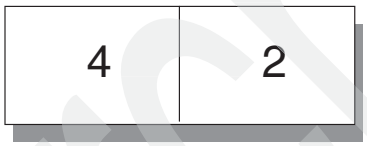


Figure 4-15 TCP: SACK-permitted option

Archive

SACK option Selective Acknowledgment (SACK) allows the receiver to inform the sender about all the segments that are received successfully. Therefore, the sender will only send the segments that got lost. If the number of the segments that have been lost since the last SACK is too large, the SACK option will be too large. As a result, the number of blocks that can be reported by the SACK option is limited to four. To reduce this, the SACK option should be used for the most recent received data. See Figure 4-16 for more details.

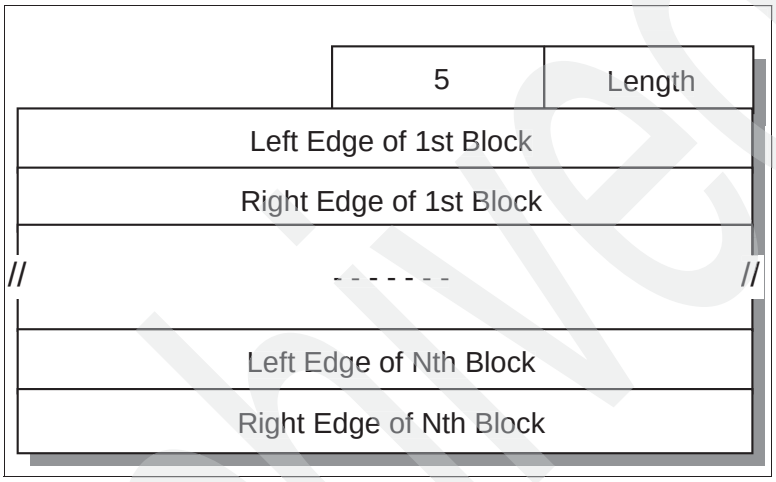


Figure 4-16 TCP: SACK option

Timestamps option The timestamps option sends a time stamp value that indicates the current value of the time stamp clock of the TCP sending the option. The Timestamp Echo Value can only be used if the ACK bit is set in the TCP header. See Figure 4-17 for more details.

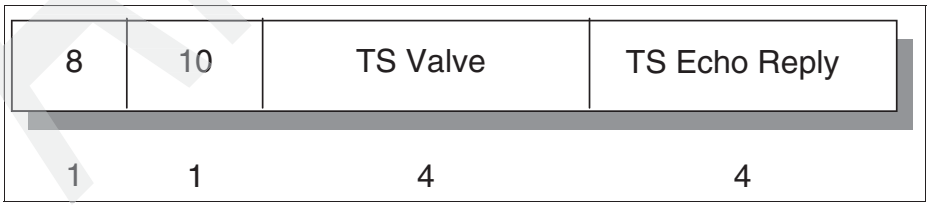


Figure 4-17 TCP: Timestamps option

Padding All zero bytes are used to fill up the TCP header to a total length that is a multiple of 32 bits.

Acknowledgments and retransmissions

TCP sends data in variable length segments. Sequence numbers are based on a byte count. Acknowledgments specify the sequence number of the next byte that the receiver expects to receive.

Consider that a segment gets lost or corrupted. In this case, the receiver will acknowledge all further well-received segments with an acknowledgment referring to the first byte of the missing packet. The sender will stop transmitting when it has sent all the bytes in the window. Eventually, a timeout will occur and the missing segment will be retransmitted.

Figure 4-18 illustrates and example where a window size of 1500 bytes and segments of 500 bytes are used.

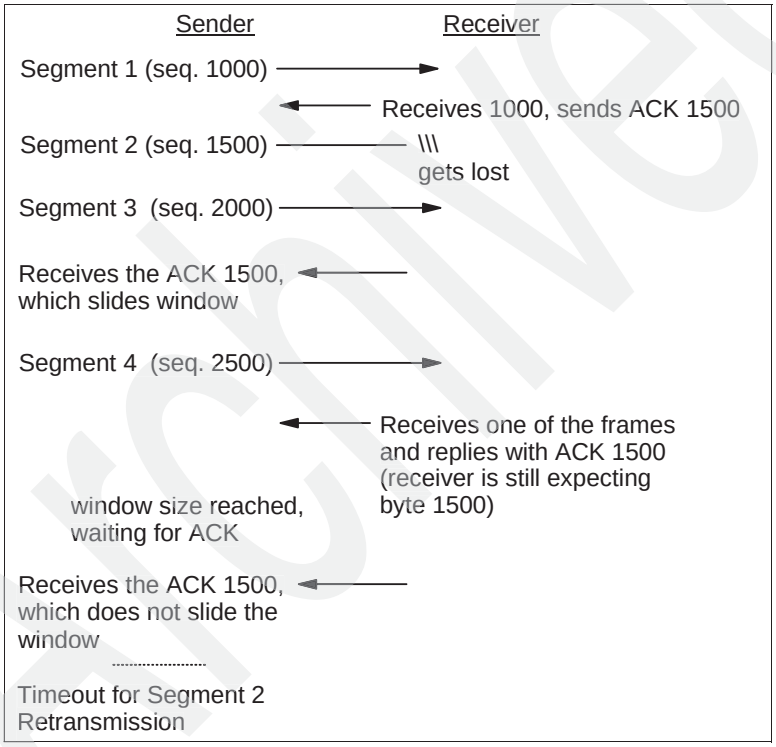


Figure 4-18 TCP: Acknowledgment and retransmission process

Archived

A problem now arises, because the sender does know that segment 2 is lost or corrupted, but does not know anything about segments 3 and 4. The sender should at least retransmit segment 2, but it could also retransmit segments 3 and 4 (because they are within the current window). It is possible that:

- Segment 3 has been received, and we do not know about segment 4. It might be received, but ACK did not reach us yet, or it might be lost.
- Segment 3 was lost, and we received the ACK 1500 on the reception of segment 4.

Each TCP implementation is free to react to a timeout as those implementing it want. It can retransmit only segment 2, but in the second case, we will be waiting again until segment 3 times out. In this case, we lose all of the throughput advantages of the window mechanism. Or TCP might immediately resend all of the segments in the current window.

Whatever the choice, maximal throughput is lost. This is because the ACK does not contain a second acknowledgment sequence number indicating the actual frame received.

Variable timeout intervals

Each TCP should implement an algorithm to adapt the timeout values to be used for the round trip time of the segments. To do this, TCP records the time at which a segment was sent, and the time at which the ACK is received. A weighted average is calculated over several of these round trip times, to be used as a timeout value for the next segment or segments to be sent.

This is an important feature, because delays can vary in IP network, depending on multiple factors, such as the load of an intermediate low-speed network or the saturation of an intermediate IP gateway.

Establishing a TCP connection

Before any data can be transferred, a connection has to be established between the two processes. One of the processes (usually the server) issues a *passive OPEN* call, the other an *active OPEN* call. The passive OPEN call remains dormant until another process tries to connect to it by an active OPEN.

As shown in Figure 4-19, in the network, three TCP segments are exchanged.

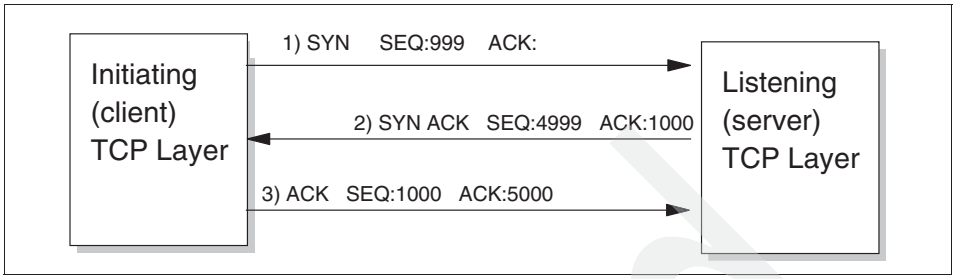


Figure 4-19 TCP: Connection establishment

This whole process is known as a three-way handshake. Note that the exchanged TCP segments include the initial sequence numbers from both sides, to be used on subsequent data transfers.

Closing the connection is done implicitly by sending a TCP segment with the FIN bit (no more data) set. Because the connection is full-duplex (that is, there are two independent data streams, one in each direction), the FIN segment only closes the data transfer in one direction. The other process will now send the remaining data it still has to transmit and also ends with a TCP segment where the FIN bit is set. The connection is deleted (status information on both sides) after the data stream is closed in both directions.

The following is a list of the different states of a TCP connection:

- LISTEN: Awaiting a connection request from another TCP layer.
- SYN-SENT: A SYN has been sent, and TCP is awaiting the response SYN.
- SYN-RECEIVED: A SYN has been received, a SYN has been sent, and TCP is awaiting an ACK.
- ESTABLISHED: The three-way handshake has been completed.
- FIN-WAIT-1: The local application has issued a CLOSE. TCP has sent a FIN, and is awaiting an ACK or a FIN.
- FIN-WAIT-2: A FIN has been sent, and an ACK received. TCP is awaiting a FIN from the remote TCP layer.
- CLOSE-WAIT: TCP has received a FIN, and has sent an ACK. It is awaiting a close request from the local application before sending a FIN.
- CLOSING: A FIN has been sent, a FIN has been received, and an ACK has been sent. TCP is awaiting an ACK for the FIN that was sent.
- LAST-ACK: A FIN has been received, and an ACK and a FIN have been sent. TCP is awaiting an ACK.

- TIME-WAIT: FINs have been received and ACK'd, and TCP is waiting two MSLs to remove the connection from the table.
- CLOSED: Imaginary, this indicates that a connection has been removed from the connection table.

4.3.2 TCP application programming interface

The TCP application programming interface is not fully defined. Only some base functions it should provide are described in RFC 793 – Transmission Control Protocol. As is the case with most RFCs in the TCP/IP protocol suite, a great degree of freedom is left to the implementers, thereby allowing for optimal operating system-dependent implementations, resulting in better efficiency and greater throughput.

The following function calls are described in the RFC:

- Open: To establish a connection takes several parameters, such as:
  - Active/Passive
  - Foreign socket
  - Local port number
  - Timeout value (optional)

  This returns a local connection name, which is used to reference this particular connection in all other functions.
- Send: Causes data in a referenced user buffer to be sent over the connection. Can optionally set the URGENT flag or the PUSH flag.
- Receive: Copies incoming TCP data to a user buffer.
- Close: Closes the connection; causes a push of all remaining data and a TCP segment with FIN flag set.
- Status: An implementation-dependent call that can return information, such as:
  - Local and foreign socket
  - Send and receive window sizes
  - Connection state
  - Local connection name
- Abort: Causes all pending Send and Receive operations to be aborted, and a RESET to be sent to the foreign TCP.

For full details, see RFC 793 – Transmission Control Protocol.

4.3.3 TCP congestion control algorithms

One big difference between TCP and UDP is the congestion control algorithm. The TCP congestion algorithm prevents a sender from overrunning the capacity of the network (for example, slower WAN links). TCP can adapt the sender's rate to network capacity and attempt to avoid potential congestion situations. In order to understand the difference between TCP and UDP, understanding basic TCP congestion control algorithms is very helpful.

Several congestion control enhancements have been added and suggested to TCP over the years. This is still an active and ongoing research area, but modern implementations of TCP contain four intertwined algorithms as basic Internet standards:

- Slow start
- Congestion avoidance
- Fast retransmit
- Fast recovery

Slow start

Old implementations of TCP start a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. Although this is OK when the two hosts are on the same LAN, if there are routers and slower links between the sender and the receiver, problems can arise. Some intermediate routers cannot handle it, packets get dropped, and retransmission results and performance is degraded.

The algorithm to avoid this is called slow start. It operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgments are returned by the other end. Slow start adds another window to the sender's TCP: the congestion window, called cwnd. When a new connection is established with a host on another network, the congestion window is initialized to one segment (for example, the segment size announced by the other end, or the default, typically 536 or 512).

Note: Congestion control is defined in RFC 2581. Additionally, RFC 3390 updates RFC 2581 such that TCP implementations can initialize the congestion window to between two and four segments, with an upper limit of 4 K.

Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit the lower value of the congestion window or the advertised window. The congestion window is flow control imposed by the

Archived

sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for this connection.

The sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth, although it is not exactly exponential, because the receiver might delay its ACKs, typically sending one ACK for every two segments that it receives.

At some point, the capacity of the IP network (for example, slower WAN links) can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has gotten too large. See Figure 4-20 for an overview of slow start in action.

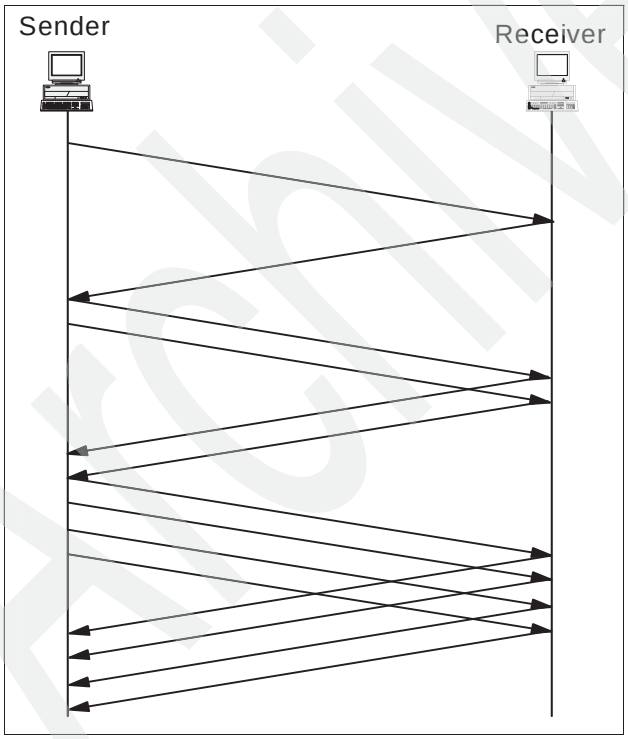


Figure 4-20 TCP: Slow start in action

Congestion avoidance

The assumption of the algorithm is that packet loss caused by damage is very small (much less than 1%). Therefore, the loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss:

- A timeout occurs.
- Duplicate ACKs are received.

Congestion avoidance and slow start are independent algorithms with different objectives. But when congestion occurs, TCP must slow down its transmission rate of packets into the network and invoke slow start to get things going again. In practice, they are implemented together.

Congestion avoidance and slow start require that two variables be maintained for each connection:

- A congestion window, cwnd
- A slow start threshold size, ssthresh

The combined algorithm operates as follows:

1. Initialization for a given connection sets cwnd to one segment and ssthresh to 65535 bytes.
2. The TCP output routine never sends more than the lower value of cwnd or the receiver's advertised window.
3. When congestion occurs (timeout or duplicate ACK), one-half of the current window size is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment.
4. When new data is acknowledged by the other end, increase cwnd, but the way it increases depends on whether TCP is performing slow start or congestion avoidance. If cwnd is less than or equal to ssthresh, TCP is in slow start; otherwise, TCP is performing congestion avoidance.

Slow start continues until TCP is halfway to where it was when congestion occurred (since it recorded half of the window size that caused the problem in step 2), and then congestion avoidance takes over. Slow start has cwnd begin at one segment, and incremented by one segment every time an ACK is received. As mentioned earlier, this opens the window exponentially: send one segment, then two, then four, and so on.

Congestion avoidance dictates that cwnd be incremented by segsize*segsize/cwnd each time an ACK is received, where segsize is the segment size and cwnd is maintained in bytes. This is a linear growth of cwnd, compared to slow start's exponential growth. The increase in cwnd should be at

most one segment each round-trip time (regardless of how many ACKs are received in that round-trip time), while slow start increments cwnd by the number of ACKs received in a round-trip time. Many implementations incorrectly add a small fraction of the segment size (typically the segment size divided by 8) during congestion avoidance. This is wrong and should not be emulated in future releases. See Figure 4-21 for an example of TCP slow start and congestion avoidance in action.

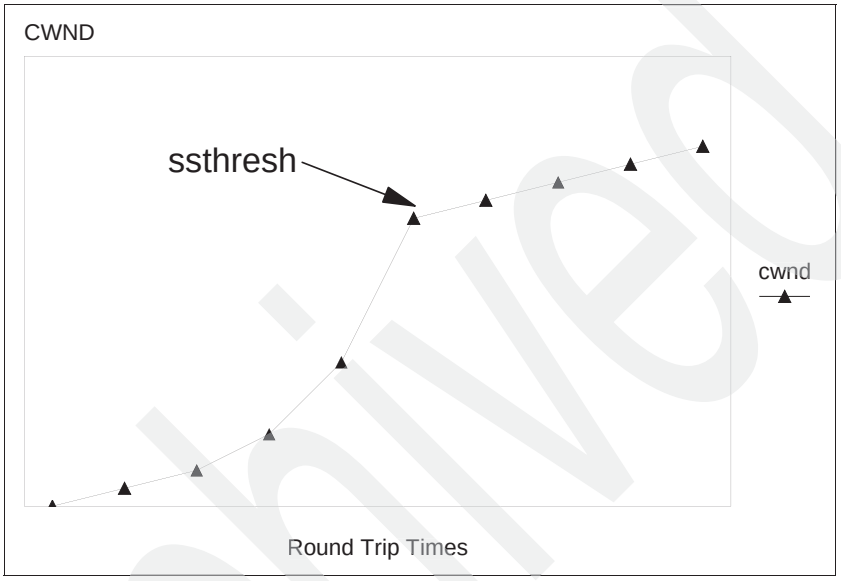


Figure 4-21 TCP: Slow start and congestion avoidance behavior in action

Fast retransmit

Fast retransmit avoids having TCP wait for a timeout to resend lost segments.

Modifications to the congestion avoidance algorithm were proposed in 1990. Before describing the change, realize that TCP can generate an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order and to tell it what sequence number is expected.

Because TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has

been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. See Figure 4-22 for an overview of TCP fast retransmit in action.

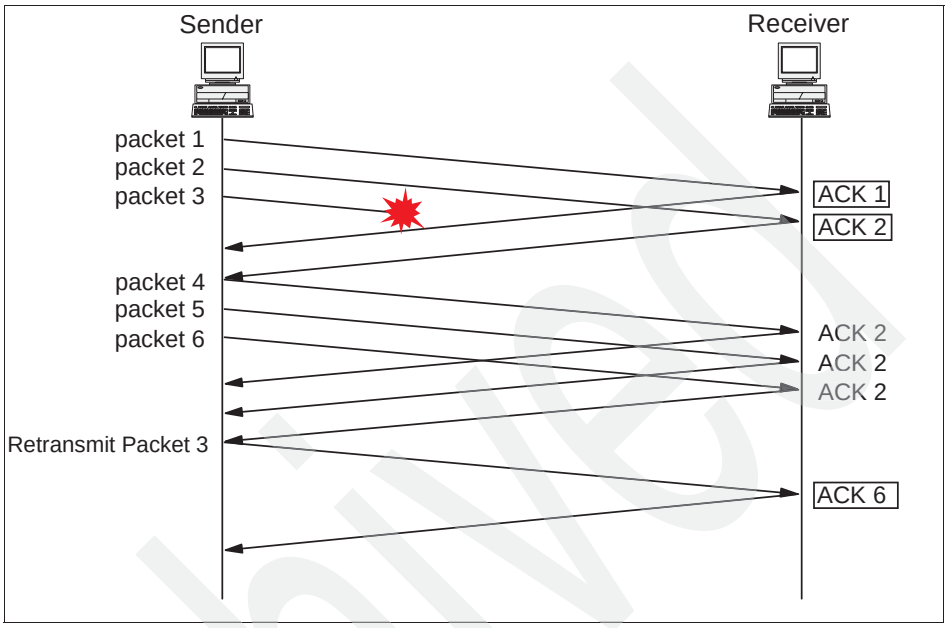


Figure 4-22 TCP: Fast retransmit in action

Fast recovery

After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start, is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows.

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Because the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start. The fast retransmit and fast recovery algorithms are usually implemented together as follows:

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus three times the segment size. This inflates the congestion window by the number of segments that have left the network and the other end has cached (3).

2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.
3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK is the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK acknowledges all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, because TCP is down to one-half the rate it was at when the packet was lost.

4.4 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 761 – DoD standard Transmission Control Protocol (January 1980)
- RFC 768 – User Datagram Protocol (August 1980)
- RFC 793 – Updated by RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP (September 2001)

Archived

5

Routing protocols

This chapter provides an overview of IP routing and discusses the various routing protocols used.

One of the basic functions provided by the IP protocol is the ability to form connections between different physical networks. A system that performs this function is called an *IP router*. This type of device attaches to two or more physical networks and forwards datagrams between the networks.

When sending data to a remote destination, a host passes datagrams to a local router. The router forwards the datagrams toward the final destination. They travel from one router to another until they reach a router connected to the destination's LAN segment. Each router along the end-to-end path selects the *next hop* device used to reach the destination. The next hop represents the next device along the path to reach the destination. It is located on a physical network connected to this intermediate system. Because this physical network differs from the one on which the system originally received the datagram, the intermediate host has *forwarded* (that is, routed) the IP datagram from one physical network to another.

© Copyright IBM Corp. 1989-2006. All rights reserved.

Figure 5-1 shows an environment where Host C is positioned to forward packets between network X and network Y.

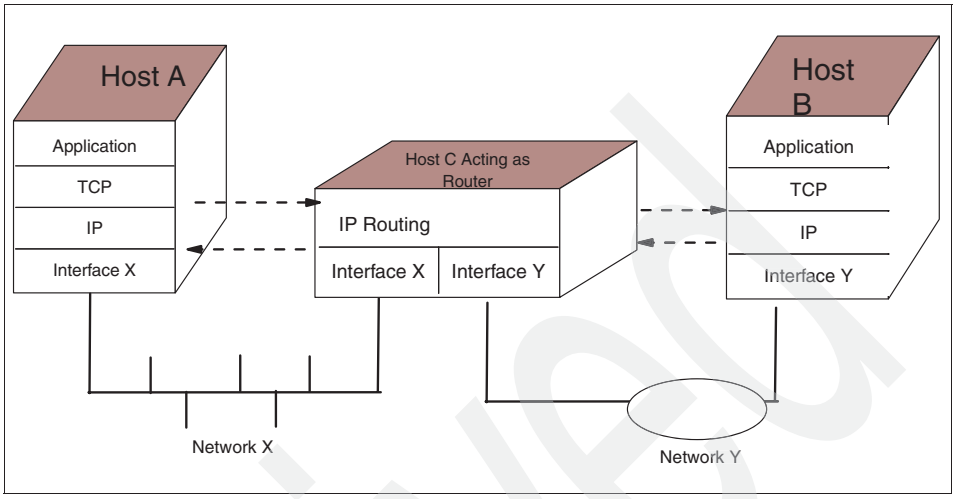


Figure 5-1 IP routing operations

The IP routing table in each device is used to forward packets between network segments. The basic table contains information about a router's locally connected networks. The configuration of the device can be extended to contain information detailing remote networks. This information provides a more complete view of the overall environment.

A robust routing protocol provides the ability to dynamically build and manage the information in the IP routing table. As network topology changes occur, the routing tables are updated with minimal or no manual intervention. This chapter details several IP routing protocols and how each protocol manages this information.

Note: In other sections of this book, the position of each protocol within the layered model of the OSI protocol stack is shown. The routing function is included as part of the internetwork layer. However, the primary function of a routing protocol is to exchange routing information with other routers. In this respect, routing protocols behave more like an application protocol. Therefore, this chapter makes no attempt to represent the position of these protocols within the overall protocol stack.

Note: Early IP routing documentation often referred to an IP router as an *IP gateway*.

5.1 Autonomous systems

The definition of an autonomous system (AS) is integral to understanding the function and scope of a routing protocol. An AS is defined as a logical portion of a larger IP network. An AS normally consists of an internetwork within an organization. It is administered by a single management authority. As shown in Figure 5-2, an AS can connect to other autonomous systems managed by the same organization. Alternatively, it can connect to other public or private networks.

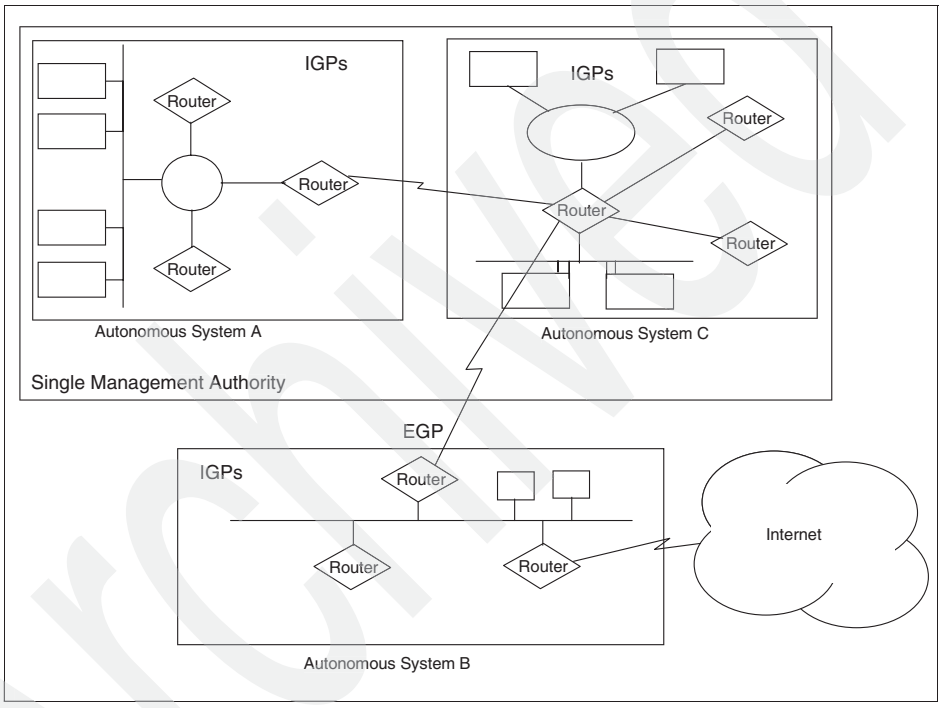


Figure 5-2 Autonomous systems

Some routing protocols are used to determine routing paths within an AS. Others are used to interconnect a set of autonomous systems:

- Interior Gateway Protocols (IGPs): Interior Gateway Protocols allow routers to exchange information within an AS. Examples of these protocols are Open Short Path First (OSPF) and Routing Information Protocol (RIP).
- Exterior Gateway Protocols (EGPs): Exterior Gateway Protocols allow the exchange of summary information between autonomous systems. An example of this type of routing protocol is Border Gateway Protocol (BGP).

Figure 5-2 on page 173 depicts the interaction between Interior and Exterior Gateway Protocols. It shows the Interior Gateway Protocols used to maintain routing information within each AS. The figure also shows the Exterior Gateway Protocols maintaining the routing information between autonomous systems.

Within an AS, multiple interior routing processes can be used. When this occurs, the AS must appear to other autonomous systems as having a single coherent interior routing plan. The AS must present a consistent view of the internal destinations.

5.2 Types of IP routing and IP routing algorithms

Routing algorithms build and maintain the IP routing table on a device. There are two primary methods used to build the routing table:

- Static routing: Static routing uses preprogrammed definitions representing paths through the network.
- Dynamic routing: Dynamic routing algorithms allow routers to automatically discover and maintain awareness of the paths through the network. This automatic discovery can use a number of currently available dynamic routing protocols. The difference between these protocols is the way they discover and calculate new routes to destination networks. They can be classified into four broad categories:
  - Distance vector protocols
  - Link state protocols
  - Path vector protocols
  - Hybrid protocols

The remainder of this section describes the operation of each algorithm.

There are several reasons for the multiplicity of protocols:

- Routing within a network and routing between networks typically have different requirements for security, stability, and scalability. Different routing protocols have been developed to address these requirements.
- New protocols have been developed to address the observed deficiencies in established protocols.
- Different-sized networks can use different routing algorithms. Small to medium-sized networks often use routing protocols that reflect the simplicity of the environment.

However, these protocols do not scale to support large, interconnected networks. More complex routing algorithms are required to support these environments.

5.2.1 Static routing

Static routing is manually performed by the network administrator. The administrator is responsible for discovering and propagating routes through the network. These definitions are manually programmed in every routing device in the environment.

After a device has been configured, it simply forwards packets out the predetermined ports. There is no communication between routers regarding the current topology of the network.

In small networks with minimal redundancy, this process is relatively simple to administer. However, there are several disadvantages to this approach for maintaining IP routing tables:

- Static routes require a considerable amount of coordination and maintenance in non-trivial network environments.
- Static routes cannot dynamically adapt to the current operational state of the network. If a destination subnetwork becomes unreachable, the static routes pointing to that network remain in the routing table. Traffic continues to be forwarded toward that destination. Unless the network administrator updates the static routes to reflect the new topology, traffic is unable to use any alternate paths that may exist.

Normally, static routes are used only in simple network topologies. However, there are additional circumstances when static routing can be attractive. For example, static routes can be used:

- To manually define a default route. This route is used to forward traffic when the routing table does not contain a more specific route to the destination.
- To define a route that is not automatically advertised within a network.
- When utilization or line tariffs make it undesirable to send routing advertisement traffic through lower-capacity WAN connections.
- When complex routing policies are required. For example, static routes can be used to guarantee that traffic destined for a specific host traverses a designated network path.
- To provide a more secure network environment. The administrator is aware of all subnetworks defined in the environment. The administrator specifically authorizes all communication permitted between these subnetworks.

- To provide more efficient resource utilization. This method of routing table management requires no network bandwidth to advertise routes between neighboring devices. It also uses less processor memory and CPU cycles to calculate network paths.

5.2.2 Distance vector routing

Distance vector algorithms are examples of dynamic routing protocols. These algorithms allow each device in the network to automatically build and maintain a local IP routing table.

The principle behind distance vector routing is simple. Each router in the internetwork maintains the *distance* or *cost* from itself to every known destination. This value represents the overall desirability of the path. Paths associated with a smaller cost value are more attractive to use than paths associated with a larger value. The path represented by the smallest cost becomes the preferred path to reach the destination.

This information is maintained in a *distance vector table*. The table is periodically advertised to each neighboring router. Each router processes these advertisements to determine the best paths through the network.

The main advantage of distance vector algorithms is that they are typically easy to implement and debug. They are very useful in small networks with limited redundancy. However, there are several disadvantages with this type of protocol:

- During an adverse condition, the length of time for every device in the network to produce an accurate routing table is called the *convergence time*. In large, complex internetworks using distance vector algorithms, this time can be excessive. While the routing tables are converging, networks are susceptible to inconsistent routing behavior. This can cause routing loops or other types of unstable packet forwarding.
- To reduce convergence time, a limit is often placed on the maximum number of hops contained in a single route. Valid paths exceeding this limit are not usable in distance vector networks.
- Distance vector routing tables are periodically transmitted to neighboring devices. They are sent even if no changes have been made to the contents of the table. This can cause noticeable periods of increased utilization in reduced capacity environments.

Enhancements to the basic distance vector algorithm have been developed to reduce the convergence and instability exposures. We describe these enhancements in 5.3.5, “Convergence and counting to infinity” on page 185.

RIP is a popular example of a distance vector routing protocol.

5.2.3 Link state routing

The growth in the size and complexity of networks in recent years has necessitated the development of more robust routing algorithms. These algorithms address the shortcoming observed in distance vector protocols.

These algorithms use the principle of a *link state* to determine network topology. A link state is the description of an interface on a router (for example, IP address, subnet mask, type of network) and its relationship to neighboring routers. The collection of these link states forms a link state database.

The process used by link state algorithms to determine network topology is straightforward:

1. Each router identifies all other routing devices on the directly connected networks.
2. Each router advertises a list of all directly connected network links and the associated cost of each link. This is performed through the exchange of link state advertisements (LSAs) with other routers in the network.
3. Using these advertisements, each router creates a database detailing the current network topology. The topology database in each router is identical.
4. Each router uses the information in the topology database to compute the most desirable routes to each destination network. This information is used to update the IP routing table.

Shortest-Path First (SPF) algorithm

The SPF algorithm is used to process the information in the topology database. It provides a tree-representation of the network. The device running the SPF algorithm is the root of the tree. The output of the algorithm is the list of shortest-paths to each destination network. Figure 5-3 on page 178 provides an example of the shortest-path algorithm executed on router A.

Archive

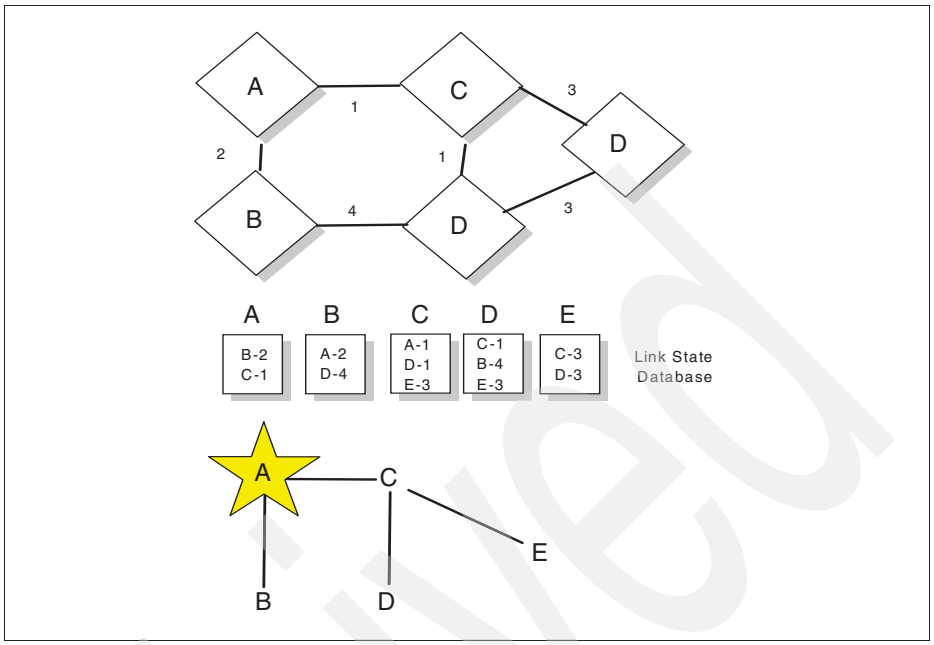


Figure 5-3 Shortest-Path First (SPF) example

Because each router is processing the same set of LSAs, each router creates an identical link state database. However, because each device occupies a different place in the network topology, the application of the SPF algorithm produces a different tree for each router.

The OSPF protocol is a popular example of a link state routing protocol.

5.2.4 Path vector routing

Path vector routing is discussed in RFC 1322; the following paragraphs are based on the RFC.

The path vector routing algorithm is somewhat similar to the distance vector algorithm in the sense that each border router advertises the destinations it can reach to its neighboring router. However, instead of advertising networks in terms of a destination and the distance to that destination, networks are advertised as destination addresses and path descriptions to reach those destinations.

A route is defined as a pairing between a destination and the attributes of the path to that destination, thus the name, path vector routing, where the routers receive a vector that contains paths to a set of destinations.

The path, expressed in terms of the domains (or confederations) traversed so far, is carried in a special path attribute that records the sequence of routing domains through which the reachability information has passed. The path represented by the smallest number of domains becomes the preferred path to reach the destination.

The main advantage of a path vector protocol is its flexibility. There are several other advantages regarding using a path vector protocol:

- The computational complexity is smaller than that of the link state protocol. The path vector computation consists of evaluating a newly arrived route and comparing it with the existing one, while conventional link state computation requires execution of an SPF algorithm.
- Path vector routing does not require all routing domains to have homogeneous policies for route selection; route selection policies used by one routing domain are not necessarily known to other routing domains. The support for heterogeneous route selection policies has serious implications for the computational complexity. The path vector protocol allows each domain to make its route selection autonomously, based only on local policies. However, path vector routing can accommodate heterogeneous route selection with little additional cost.
- Only the domains whose routes are affected by the changes have to recompute.
- Suppression of routing loops is implemented through the path attribute, in contrast to link state and distance vector, which use a globally-defined monotonically thereby increasing metric for route selection. Therefore, different confederation definitions are accommodated because looping is avoided by the use of full path information.
- Route computation precedes routing information dissemination. Therefore, only routing information associated with the routes selected by a domain is distributed to adjacent domains.
- Path vector routing has the ability to selectively hide information.

However, there are disadvantages to this approach, including:

- Topology changes only result in the recomputation of routes affected by these changes, which is more efficient than complete recomputation. However, because of the inclusion of full path information with each distance vector, the effect of a topology change can propagate farther than in traditional distance vector algorithms.

- Unless the network topology is fully meshed or is able to appear so, routing loops can become an issue.

BGP is a popular example of a path vector routing protocol.

5.2.5 Hybrid routing

The last category of routing protocols is hybrid protocols. These protocols attempt to combine the positive attributes of both distance vector and link state protocols. Like distance vector, hybrid protocols use metrics to assign a preference to a route. However, the metrics are more accurate than conventional distance vector protocols. Like link state algorithms, routing updates in hybrid protocols are event driven rather than periodic. Networks using hybrid protocols tend to converge more quickly than networks using distance vector protocols. Finally, these protocols potentially reduce the costs of link state updates and distance vector advertisements.

Although open hybrid protocols exist, this category is almost exclusively associated with the proprietary EIGRP algorithm. EIGRP was developed by Cisco Systems, Inc.

5.3 Routing Information Protocol (RIP)

RIP is an example of an interior gateway protocol designed for use within small autonomous systems. RIP is based on the Xerox XNS routing protocol. Early implementations of RIP were readily accepted because the code was incorporated in the Berkeley Software Distribution (BSD) UNIX-based operating system. RIP is a distance vector protocol.

In mid-1988, the IETF issued RFC 1058 with updates in RFC2453, which describes the standard operations of a RIP system. However, the RFC was issued after many RIP implementations had been completed. For this reason, some RIP systems do not support the entire set of enhancements to the basic distance vector algorithm (for example, poison reverse and triggered updates).

5.3.1 RIP packet types

The RIP protocol specifies two packet types. These packets can be sent by any device running the RIP protocol:

- Request packets: A request packet queries neighboring RIP devices to obtain their distance vector table. The request indicates if the neighbor should return either a specific subset or the entire contents of the table.

- Response packets: A response packet is sent by a device to advertise the information maintained in its local distance vector table. The table is sent during the following situations:
  - The table is automatically sent every 30 seconds.
  - The table is sent as a response to a request packet generated by another RIP node.
  - If triggered updates are supported, the table is sent when there is a change to the local distance vector table. We discuss triggered updates in “Triggered updates” on page 188.

When a response packet is received by a device, the information contained in the update is compared against the local distance vector table. If the update contains a lower cost route to a destination, the table is updated to reflect the new path.

5.3.2 RIP packet format

RIP uses a specific packet format to share information about the distances to known network destinations. RIP packets are transmitted using UDP datagrams. RIP sends and receives datagrams using UDP port 520.

RIP datagrams have a maximum size of 512 octets. Updates larger than this size must be advertised in multiple datagrams. In LAN environments, RIP datagrams are sent using the MAC all-stations broadcast address and an IP network broadcast address. In point-to-point or non-broadcast environments, datagrams are specifically addressed to the destination device.

Archived

The RIP packet format is shown in Figure 5-4.

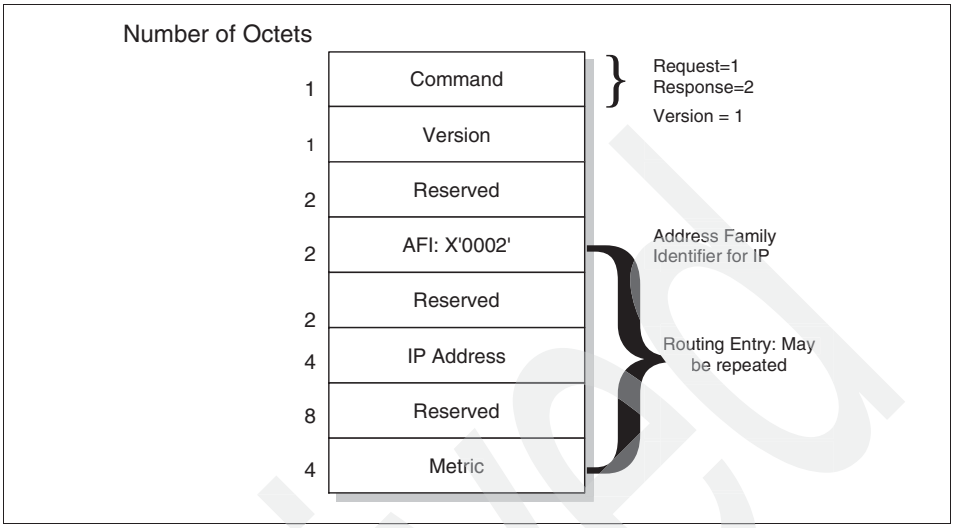


Figure 5-4 RIP packet format

A 512 byte packet size allows a maximum of 25 routing entries to be included in a single RIP advertisement.

5.3.3 RIP modes of operation

RIP hosts have two modes of operation:

- Active mode: Devices operating in active mode advertise their distance vector table and also receive routing updates from neighboring RIP hosts. Routing devices are typically configured to operate in active mode.
- Passive (or silent) mode: Devices operating in this mode simply receive routing updates from neighboring RIP devices. They do not advertise their distance vector table. End stations are typically configured to operate in passive mode.

5.3.4 Calculating distance vectors

The distance vector table describes each destination network. The entries in this table contain the following information:

- The destination network (vector) described by this entry in the table.

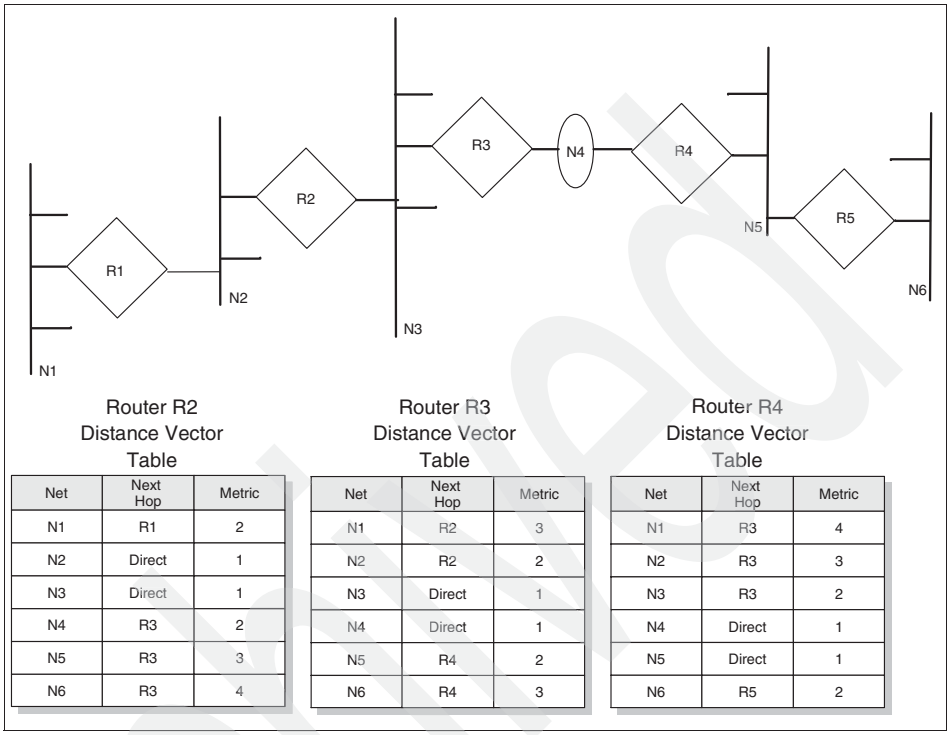
- The associated cost (distance) of the most attractive path to reach this destination. This provides the ability to differentiate between multiple paths to a destination. In this context, the terms distance and cost can be misleading. They have no direct relationship to physical distance or monetary cost.
- The IP address of the next-hop device used to reach the destination network.

Each time a routing table advertisement is received by a device, it is processed to determine if any destination can be reached by a lower cost path. This is done using the RIP distance vector algorithm. The algorithm can be summarized as:

- At router initialization, each device contains a distance vector table listing each directly attached networks and configured cost. Typically, each network is assigned a cost of 1. This represents a single hop through the network. The total number of hops in a route is equal to the total cost of the route. However, cost can be changed to reflect other measurements such as utilization, speed, or reliability.
- Each router periodically (typically every 30 seconds) transmits its distance vector table to each of its neighbors. The router can also transmit the table when a topology change occurs. Each router uses this information to update its local distance vector table:
  - The total cost to each destination is calculated by adding the cost reported in a neighbor's distance vector table to the cost of the link to that neighbor. The path with the least cost is stored in the distance vector table.
  - All updates automatically supersede the previous information in the distance vector table. This allows RIP to maintain the integrity of the routes in the routing table.
- The IP routing table is updated to reflect the least-cost path to each destination.

Archive

Figure 5-5 illustrates the distance vector tables for three routers within a simple internetwork.



Router R2 Distance Vector Table

Net	Next Hop	Metric
N1	R1	2
N2	Direct	1
N3	Direct	1
N4	R3	2
N5	R3	3
N6	R3	4

Router R3 Distance Vector Table

Net	Next Hop	Metric
N1	R2	3
N2	R2	2
N3	Direct	1
N4	Direct	1
N5	R4	2
N6	R4	3

Router R4 Distance Vector Table

Net	Next Hop	Metric
N1	R3	4
N2	R3	3
N3	R3	2
N4	Direct	1
N5	Direct	1
N6	R5	2

Figure 5-5 A sample distance vector routing table

Archived

5.3.5 Convergence and counting to infinity

Given sufficient time, this algorithm will correctly calculate the distance vector table on each device. However, during this convergence time, erroneous routes may propagate through the network. Figure 5-6 shows this problem.

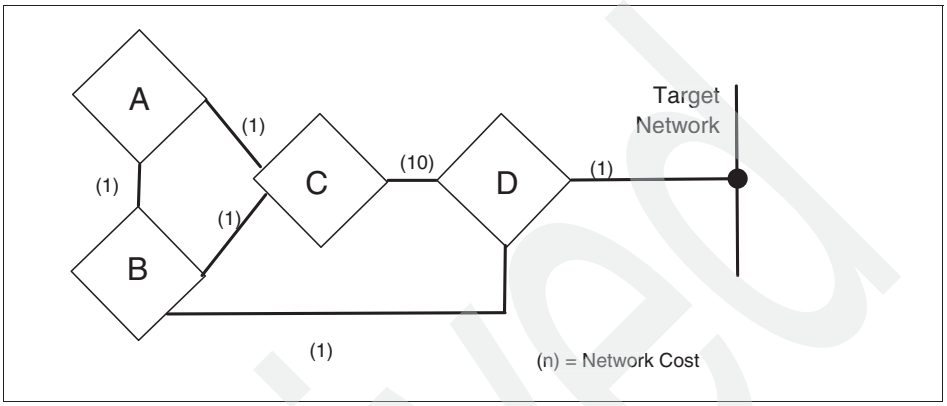


Figure 5-6 Counting to infinity sample network

This network contains four interconnected routers. Each link has a cost of 1, except for the link connecting router C and router D; this link has a cost of 10. The costs have been defined so that forwarding packets on the link connecting router C and router D is undesirable. After the network has converged, each device has routing information describing all networks.

For example, to reach the target network, the routers have the following information:

- Router D to the target network: Directly connected network. Metric is 1.
- Router B to the target network: Next hop is router D. Metric is 2.
- Router C to the target network: Next hop is router B. Metric is 3.
- Router A to the target network: Next hop is router B. Metric is 3.

Consider an adverse condition where the link connecting router B and router D fails. After the network has reconverged, all routes use the link connecting router C and router D to reach the target network. However, this reconvergence time

can be considerable. Figure 5-7 illustrates how the routes to the target network are updated throughout the reconvergence period. For simplicity, this figure assumes all routers send updates at the same time.

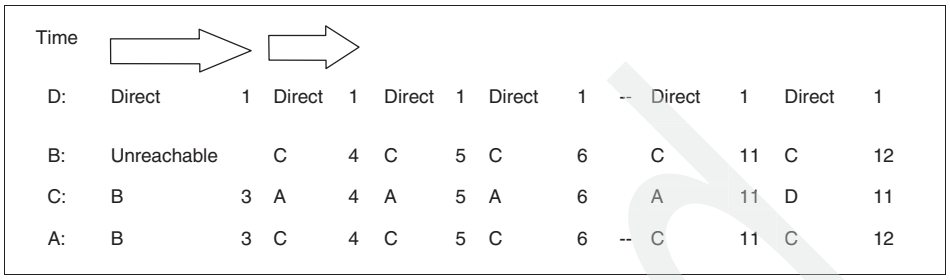


Figure 5-7 Network convergence sequence

Reconvergence begins when router B notices that the route to router D is unavailable. Router B is able to immediately remove the failed route because the link has timed out. However, a considerable amount of time passes before the other routers remove their references to the failed route. This is described in the sequence of updates shown in Figure 5-7:

1. Prior to the adverse condition occurring, router A and router C have a route to the target network through router B.
2. The adverse condition occurs when the link connecting router D and router B fails. Router B recognizes that its preferred path to the target network is now invalid.
3. Router A and router C continue to send updates reflecting the route through router B. This route is actually invalid because the link connecting router D and router B has failed.
4. Router B receives the updates from router A and router C. Router B believes it should now route traffic to the target network through either router A or router C. In reality, this is not a valid route, because the routes in router A and router C are vestiges of the previous route through router B.
5. Using the routing advertisement sent by router B, router A and router C are able to determine that the route through router B has failed. However, router A and router C now believe the preferred route exists through the partner.

Network convergence continues as router A and router C engage in an extended period of mutual deception. Each device claims to be able to reach the target network through the partner device. The path to reach the target network now contains a routing loop.

The manner in which the costs in the distance vector table increment gives rise to the term *counting to infinity*. The costs continues to increment, theoretically to

infinity. To minimize this exposure, whenever a network is unavailable, the incrementing of metrics through routing updates must be halted as soon as it is practical to do so. In a RIP environment, costs continue to increment until they reach a maximum value of 16. This limit is defined in RFC 1058.

A side effect of the metric limit is that it also limits the number of hops a packet can traverse from source network to destination network. In a RIP environment, any path exceeding 15 hops is considered invalid. The routing algorithm will discard these paths.

There are two enhancements to the basic distance vector algorithm that can minimize the counting to infinity problem:

- Split horizon with poison reverse
- Triggered updates

These enhancements do not impact the maximum metric limit.

Split horizon

The excessive convergence time caused by counting to infinity can be reduced with the use of split horizon. This rule dictates that routing information is prevented from exiting the router on an interface through which the information was received.

The basic split horizon rule is not supported in RFC 1058. Instead, the standard specifies the enhanced split horizon with poison reverse algorithm. The basic rule is presented here for background and completeness. The enhanced algorithm is reviewed in the next section.

The incorporation of split horizon modifies the sequence of routing updates shown in Figure 5-7 on page 186. The new sequence is shown in Figure 5-8. The tables show that convergence occurs considerably faster using the split horizon rule.

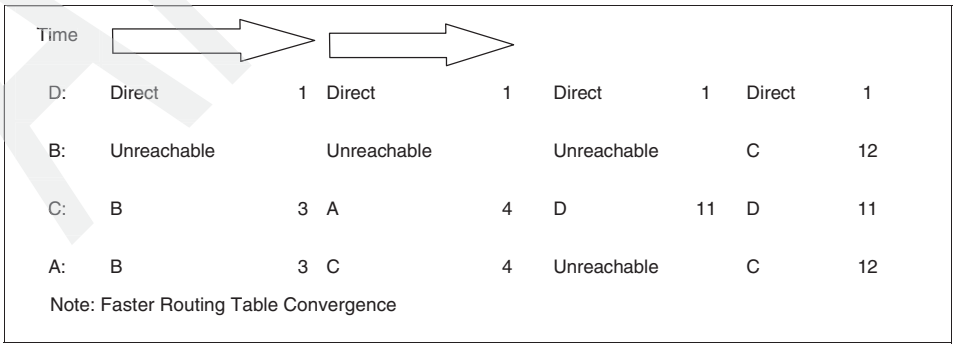


Figure 5-8 Network convergence with split horizon

The limitation to this rule is that each node must wait for the route to the unreachable destination to time out before the route is removed from the distance vector table. In RIP environments, this timeout is at least three minutes after the initial outage. During that time, the device continues to provide erroneous information to other nodes about the unreachable destination. This propagates routing loops and other routing anomalies.

Split horizon with poison reverse

Poison reverse is an enhancement to the standard split horizon implementation. It is supported in RFC 1058. With poison reverse, all known networks are advertised in each routing update. However, those networks learned through a specific interface are advertised as unreachable in the routing announcements sent out to that interface.

This drastically improves convergence time in complex, highly-redundant environments. With poison reverse, when a routing update indicates that a network is unreachable, routes are immediately removed from the routing table. This breaks erroneous, looping routes before they can propagate through the network. This approach differs from the basic split horizon rule where routes are eliminated through timeouts.

Poison reverse has no benefit in networks with no redundancy (single path networks).

One disadvantage to poison reverse is that it might significantly increase the size of routing annoucements exchanged between neighbors. This is because all routes in the distance vector table are included in each announcement. Although this is generally not an issue on local area networks, it can cause periods of increased utilization on lower-capacity WAN connections.

Triggered updates

Like split horizon with poison reverse, algorithms implementing triggered updates are designed to reduce network convergence time. With triggered updates, whenever a router changes the cost of a route, it immediately sends the modified distance vector table to neighboring devices. This mechanism ensures that topology change notifications are propagated quickly, rather than at the normal periodic interval.

Triggered updates are supported in RFC 1058.

5.3.6 RIP limitations

There are a number of limitations observed in RIP environments:

- Path cost limits: The resolution to the counting to infinity problem enforces a maximum cost for a network path. This places an upper limit on the maximum network diameter. Networks requiring paths greater than 15 hops must use an alternate routing protocol.
- Network-intensive table updates: Periodic broadcasting of the distance vector table can result in increased utilization of network resources. This can be a concern in reduced-capacity segments.
- Relatively slow convergence: RIP, like other distance vector protocols, is relatively slow to converge. The algorithms rely on timers to initiate routing table advertisements.
- No support for variable length subnet masking: Route advertisements in a RIP environment do not include subnet masking information. This makes it impossible for RIP networks to deploy variable length subnet masks.

5.4 Routing Information Protocol Version 2 (RIP-2)

The IETF recognizes two versions of RIP:

- RIP Version 1 (RIP-1): This protocol is described in RFC 1058.
- RIP Version 2 (RIP-2): RIP-2 is also a distance vector protocol designed for use within an AS. It was developed to address the limitations observed in RIP-1. RIP-2 is described in RFC 2453. The standard (STD 56) was published in late 1994.

In practice, the term RIP refers to RIP-1. Whenever you encounter the term RIP in TCP/IP literature, it is safe to assume that the reference is to RIP Version 1 unless otherwise stated. This same convention is used in this document. However, when the two versions are being compared, the term RIP-1 is used to avoid confusion.

RIP-2 is similar to RIP-1. It was developed to extend RIP-1 functionality in small networks. RIP-2 provides these additional benefits not available in RIP-1:

- Support for CIDR and VLSM: RIP-2 supports supernetting (that is, CIDR) and variable-length subnet masking. This support was the major reason the new standard was developed. This enhancement positions the standard to accommodate a degree of addressing complexity not supported in RIP-1.

- Support for multicasting: RIP-2 supports the use of multicasting rather than simple broadcasting of routing annoucements. This reduces the processing load on hosts not listening for RIP-2 messages. To ensure interoperability with RIP-1 environments, this option is configured on each network interface.
- Support for authentication: RIP-2 supports authentication of any node transmitting route advertisements. This prevents fraudulent sources from corrupting the routing table.
- Support for RIP-1: RIP-2 is fully interoperable with RIP-1. This provides backward-compatibility between the two standards.

As noted in the RIP-1 section, one notable shortcoming in the RIP-1 standard is the implementation of the metric field. RIP-1 specifies the metric as a value between 0 and 16. To ensure compatibility with RIP-1 networks, RIP-2 preserves this definition. In both standards, networks paths with a hop-count greater than 15 are interpreted as unreachable.

5.4.1 RIP-2 packet format

The original RIP-1 specification was designed to support future enhancements. The RIP-2 standard was able to capitalize on this feature. RIP-2 developers noted that a RIP-1 packet already contains a version field and that 50% of the octets are unused.

Archived

Figure 5-9 illustrates the contents of a RIP-2 packet. The packet is shown with authentication information. The first entry in the update contains either a routing entry or an authentication entry. If the first entry is an authentication entry, 24 additional routing entries can be included in the message. If there is no authentication information, 25 routing entries can be provided.

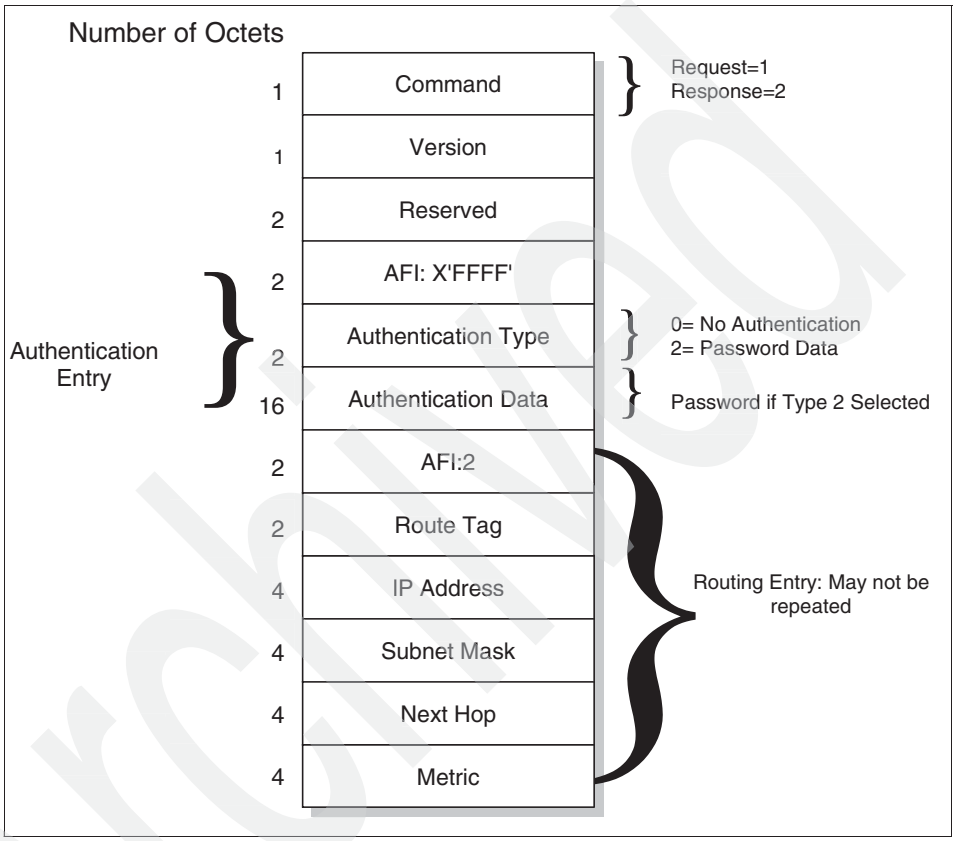


Figure 5-9 RIP-2 packet format

The use of the command field, IP address field, and metric field in a RIP-2 message is identical to the use in a RIP-1 message. Otherwise, the changes implemented in a RIP-2 packets include:

Version The value contained in this field must be two. This instructs RIP-1 routers to ignore any information contained in the previously unused fields.

AFI (Address Family) A value of x'0002' indicates the address contained in the network address field is an IP address. An value of x'FFFF' indicates an authentication entry.

Authentication Type This field defines the remaining 16 bytes of the authentication entry. A value of 0 indicates *no* authentication. A value of two indicates the authentication data field contains password data.

Authentication Data This field contains a 16-byte password.

Route Tag This field is intended to differentiate between internal and external routes. Internal routes are learned through RIP-2 within the same network or AS.

Subnet Mask This field contains the subnet mask of the referenced network.

Next Hop This field contains a recommendation about the next hop the router should use when sending datagrams to the referenced network.

5.4.2 RIP-2 limitations

RIP-2 was developed to address many of the limitations observed in RIP-1. However, the path cost limits and slow convergence inherent in RIP-1 networks are also concerns in RIP-2 environments.

In addition to these concerns, there are limitations to the RIP-2 authentication process. The RIP-2 standard does not encrypt the authentication password. It is transmitted in clear text. This makes the network vulnerable to attack by anyone with direct physical access to the environment.

5.5 RIPng for IPv6

RIPng was developed to allow routers within an IPv6-based network to exchange information used to compute routes. It is documented in RFC 2080. We provide additional information regarding IPv6 in 9.1, “IPv6 introduction” on page 328.

Like the other protocols in the RIP family, RIPng is a distance vector protocol designed for use within a small autonomous system. RIPng uses the same algorithms, timers, and logic used in RIP-2.

RIPng has many of the same limitations inherent in other distance vector protocols. Path cost restrictions and convergence time remain a concern in RIPng networks.

5.5.1 Differences between RIPng and RIP-2

There are two important distinctions between RIP-2 and RIPng:

- Support for authentication: The RIP-2 standard includes support for authenticating a node transmitting routing information. RIPng does not include any native authentication support. Rather, RIPng uses the security features inherent in IPv6. In addition to authentication, these security features provide the ability to encrypt each RIPng packet. This can control the set of devices that receive the routing information.One consequence of using IPv6 security features is that the AFI field within the RIPng packet is eliminated. There is no longer a need to distinguish between authentication entries and routing entries within an advertisement.
- Support for IPv6 addressing formats: The fields contained in RIPng packets were updated to support the longer IPv6 address format.

5.5.2 RIPng packet format

RIPng packets are transmitted using UDP datagrams. RIPng sends and receives datagrams using UDP port number 521.

The format of a RIPng packet is similar to the RIP-2 format. Specifically, both packets contain a 4 octet command header followed by a set of 20 octet route entries. The RIPng packet format is shown in Figure 5-10.

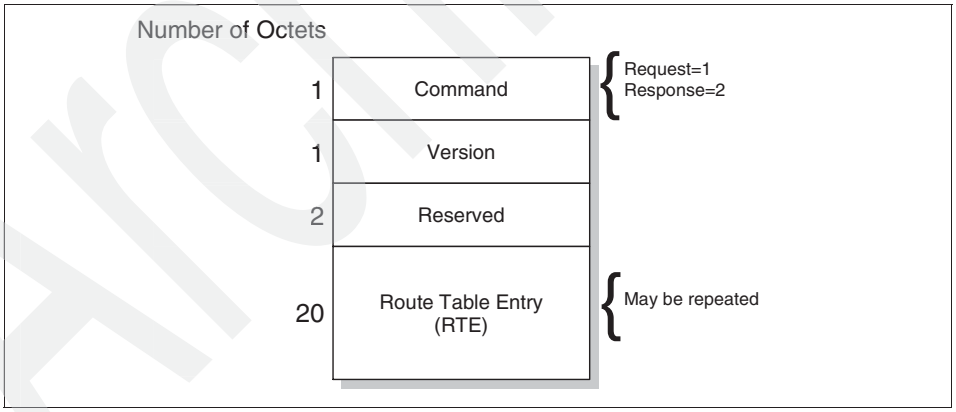


Figure 5-10 RIPng packet format

Archived

The use of the command field and the version field is identical to the use in a RIP-2 packet. However, the fields containing routing information have been updated to accommodate the 16 octet IPv6 address. These fields are used differently than the corresponding fields in a RIP-1 or RIP-2 packet. The format of the RTE is shown in Figure 5-11.

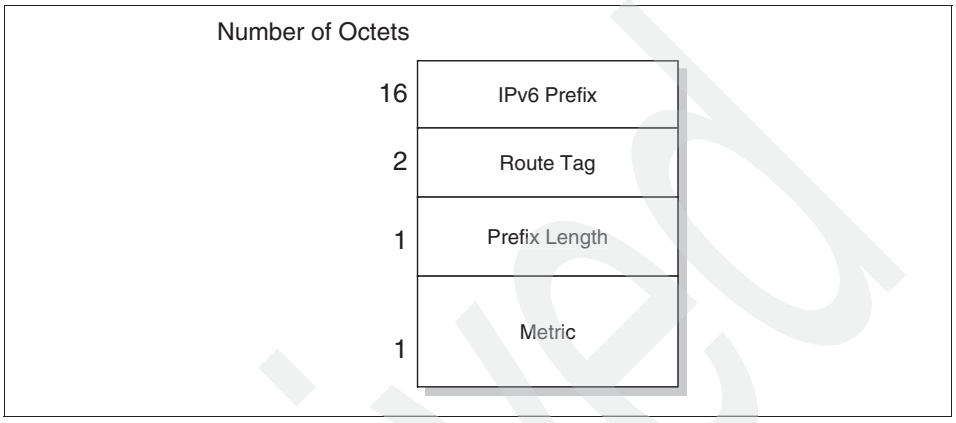


Figure 5-11 Route table entry (RTE)

In RIPng, the combination of the IP prefix and the prefix length identifies the route to be advertised. The metric remains encoded in a 1 octet field. This length is sufficient because RIPng uses a maximum hop-count of 16.

Another difference between RIPng and RIP-2 is the process used to determine the next hop. In RIP-2, each route table entry contains a next hop field. In RIPng, including this information in each RTE would have doubled the size of the advertisement. Therefore, in RIPng, the next hop is included in a special type of

Archived

RTE. The specified next hop applies to each subsequent routing table entry in the advertisement. The format of an RTE used to specify the next hop is shown in Figure 5-12.

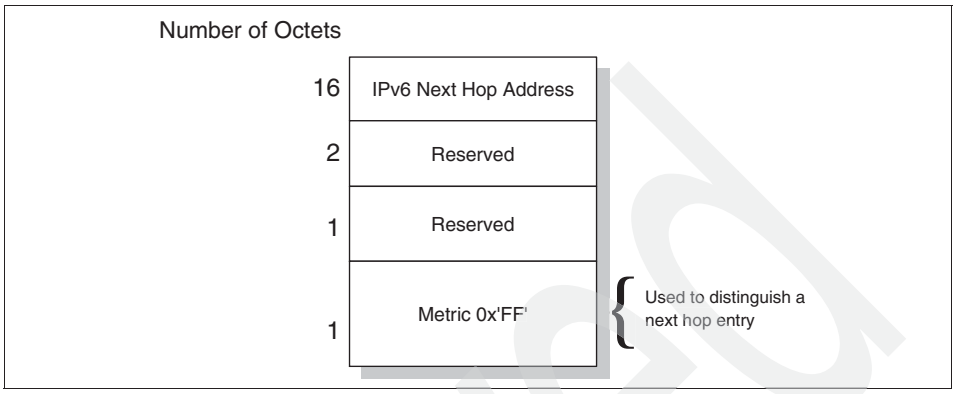


Figure 5-12 Next Hop route table entry (RTE)

The next hop RTE is identified by a value of 0x'FF' in the metric field. This reserved value is outside the valid range of metrics.

The use of RTEs and next hop RTEs is shown in Figure 5-13.

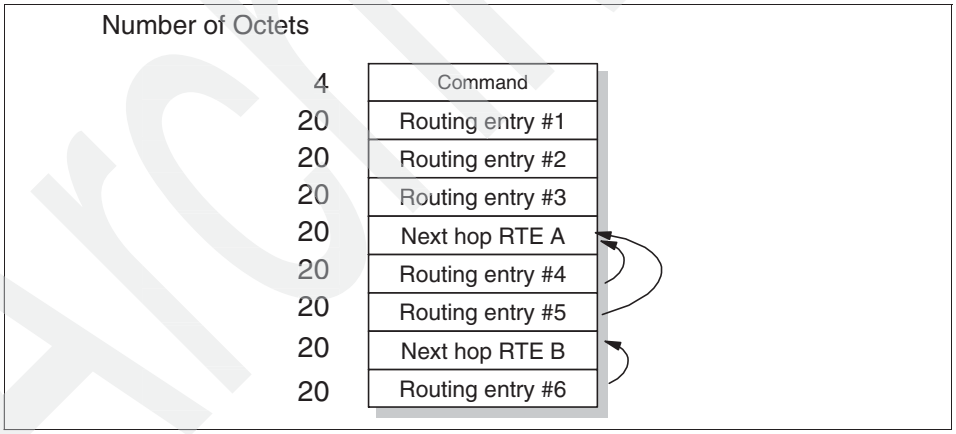


Figure 5-13 Using the RIPng RTE

In this example, the first three routing entries do not have a corresponding next hop RTE. The address prefixes specified by these entries will be routed through the advertising router. The prefixes included in routing entries 4 and 5 will route through the next hop address specified in the next hop RTE A. The prefix included in routing entry 6 will route through the next hop address specified in the next hop RTE B.

5.6 Open Shortest Path First (OSPF)

The Open Shortest Path First (OSPF) protocol is another example of an interior gateway protocol. It was developed as a non-proprietary routing alternative to address the limitations of RIP. Initial development started in 1988 and was finalized in 1991. Subsequent updates to the protocol continue to be published. The current version of the standard is documented in RFC 2328.

OSPF provides a number of features not found in distance vector protocols. Support for these features has made OSPF a widely-deployed routing protocol in large networking environments. In fact, RFC 1812 – Requirements for IPv4 Routers, lists OSPF as the only required dynamic routing protocol. The following features contribute to the continued acceptance of the OSPF standard:

- Equal cost load balancing: The simultaneous use of multiple paths can provide more efficient utilization of network resources.
- Logical partitioning of the network: This reduces the propagation of outage information during adverse conditions. It also provides the ability to aggregate routing announcements that limit the advertisement of unnecessary subnet information.
- Support for authentication: OSPF supports the authentication of any node transmitting route advertisements. This prevents fraudulent sources from corrupting the routing tables.
- Faster convergence time: OSPF provides instantaneous propagation of routing changes. This expedites the convergence time required to update network topologies.
- Support for CIDR and VLSM: This allows the network administrator to efficiently allocate IP address resources.

OSPF is a link state protocol. As with other link state protocols, each OSPF router executes the SPF algorithm (“Shortest-Path First (SPF) algorithm” on page 177) to process the information stored in the link state database. The algorithm produces a shortest-path tree detailing the preferred routes to each destination network.

5.6.1 OSPF terminology

OSPF uses specific terminology to describe the operation of the protocol.

OSPF areas

OSPF networks are divided into a collection of *areas*. An area consists of a logical grouping of networks and routers. The area can coincide with geographic or administrative boundaries. Each area is assigned a 32-bit *area ID*.

Subdividing the network provides the following benefits:

- Within an area, every router maintains an identical topology database describing the routing devices and links within the area. These routers have no knowledge of topologies outside the area. They are only aware of routes to these external destinations. This reduces the size of the topology database maintained by each router.
- Areas limit the potentially explosive growth in the number of link state updates. Most LSAs are distributed only within an area.
- Areas reduce the CPU processing required to maintain the topology database. The SPF algorithm is limited to managing changes within the area.

Backbone area and area 0

All OSPF networks contain at least one area. This area is known as area 0 or the backbone area. Additional areas can be created based on network topology or other design requirements.

In networks containing multiple areas, the backbone physically connects to all other areas. OSPF expects all areas to announce routing information directly into the backbone. The backbone then announces this information into other areas.

Figure 5-14 on page 198 depicts a network with a backbone area and four additional areas.

Archived

Intra-area, area border, and AS boundary routers

There are three classifications of routers in an OSPF network. Figure 5-14 illustrates the interaction of these devices.

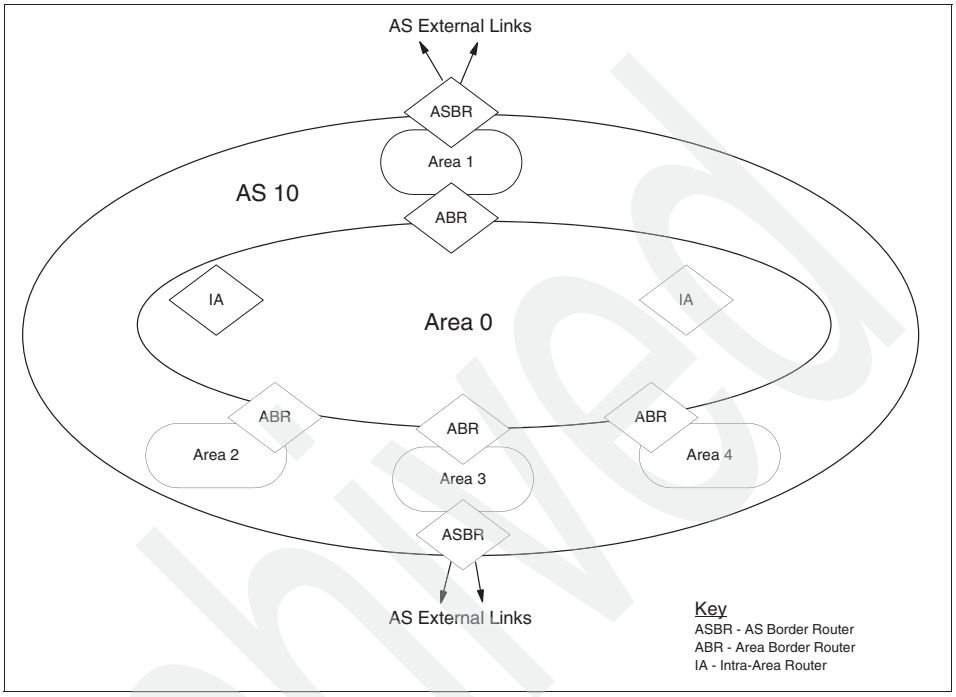


Figure 5-14 OSPF router types

Where:

Intra-area routers This class of router is logically located entirely within an OSPF area. Intra-area routers maintain a topology database for their local area.

Area border routers (ABR) This class of router is logically connected to two or more areas. One area must be the backbone area. An ABR is used to interconnect areas. They maintain a separate topology database for each attached area. ABRs also execute separate instances of the SPF algorithm for each area.

Archived

AS boundary routers (ASBR) This class of router is located at the periphery of an OSPF internetwork. It functions as a gateway exchanging reachability between the OSPF network and other routing environments.

ASBRs are responsible for announcing AS external link advertisements through the AS. We provide more information about external link advertisements in 5.6.4, “OSPF route redistribution” on page 208.

Each router is assigned a 32-bit *router ID* (RID). The RID uniquely identifies the device. One popular implementation assigns the RID from the lowest-numbered IP address configured on the router.

Physical network types

OSPF categorizes network segments into three types. The frequency and types of communication occurring between OSPF devices connected to these networks is impacted by the network type:

- Point-to-point: Point-to-point networks directly link two routers.
- Multi-access: Multi-access networks support the attachment of more than two routers.

  They are further subdivided into two types:

  – Broadcast networks have the capability of simultaneously directing a packet to all attached routers. This capability uses an address that is recognized by all devices. Ethernet and token-ring LANs are examples of OSPF broadcast multi-access networks.
  – Non-broadcast networks do not have broadcasting capabilities. Each packet must be specifically addressed to every router in the network. X.25 and frame relay networks are examples of OSPF non-broadcast multi-access networks.
- Point-to-multipoint: Point-to-multipoint networks are a special case of multi-access, non-broadcast networks. In a point-to-multipoint network, a device is not required to have a direct connection to every other device. This is known as a partially meshed environment.

Neighbor routers and adjacencies

Routers that share a common network segment establish a neighbor relationship on the segment. Routers must agree on the following information to become neighbors:

- Area ID: The routers must belong to the same OSPF area.
- Authentication: If authentication is defined, the routers must specify the same password.

- Hello and dead intervals: The routers must specify the same timer intervals used in the Hello protocol. We describe this protocol further in “OSPF packet types” on page 203.
- Stub area flag: The routers must agree that the area is configured as a stub area. We describe stub areas further in 5.6.5, “OSPF stub areas” on page 210.

After two routers have become neighbors, an adjacency relationship can be formed between the devices. Neighboring routers are considered adjacent when they have synchronized their topology databases. This occurs through the exchange of link state information.

Designated and backup designated router

The exchange of link state information between neighbors can create significant quantities of network traffic. To reduce the total bandwidth required to synchronize databases and advertise link state information, a router does not necessarily develop adjacencies with every neighboring device:

- Multi-access networks: Adjacencies are formed between an individual router and the (backup) designated router.
- Point-to-point networks: An adjacency is formed between both devices.

Each multi-access network elects a designated router (DR) and backup designated router (BDR). The DR performs two key functions on the network segment:

- It forms adjacencies with all routers on the multi-access network. This causes the DR to become the focal point for forwarding LSAs.
- It generates network link advertisements listing each router connected to the multi-access network. For additional information regarding network link advertisements, see “Link state advertisements and flooding” on page 201.

The BDR forms the same adjacencies as the designated router. It assumes DR functionality when the DR fails.

Each router is assigned an 8-bit priority, indicating its ability to be selected as the DR or BDR. A router priority of zero indicates that the router is not eligible to be selected. The priority is configured on each interface in the router.

Figure 5-15 illustrates the relationship between neighbors. No adjacencies are formed between routers that are not selected to be the DR or BDR.

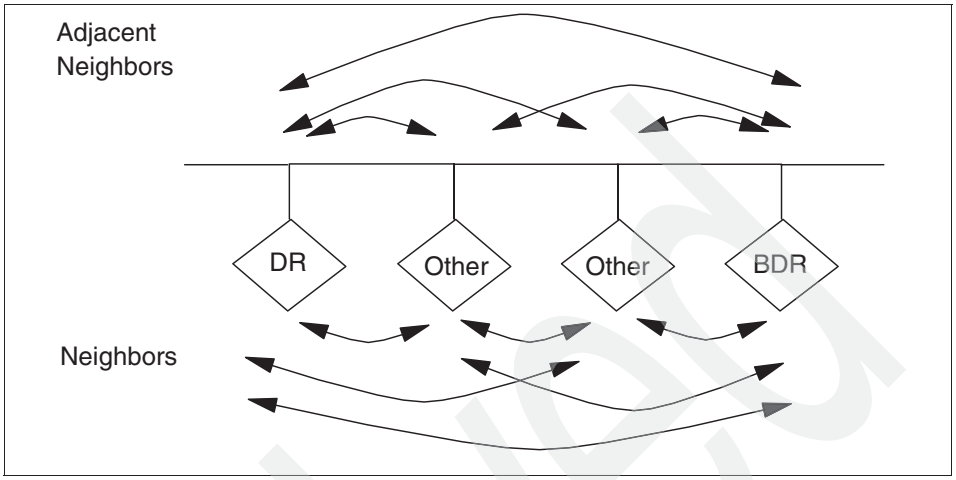


Figure 5-15 Relationship between adjacencies and neighbors

Link state database

The link state database is also called the *topology database*. It contains the set of link state advertisements describing the OSPF network and any external connections. Each router within the area maintains an identical copy of the link state database.

Note: RFC 2328 uses the term link state database in preference to topology database. The former term has the advantage in that it describes the contents of the database. The latter term is more descriptive of the purpose of the database. This book has previously used the term topology database for this reason. However for the remainder of the OSPF section, we refer to it as the link state database.

Link state advertisements and flooding

The contents of an LSA describe an individual network component (that is, router, segment, or external destination). LSAs are exchanged between adjacent OSPF routers. This is done to synchronize the link state database on each device.

When a router generates or modifies an LSA, it must communicate this change throughout the network. The router starts this process by forwarding the LSA to each adjacent device. Upon receipt of the LSA, these neighbors store the information in their link state database and communicate the LSA to their

neighbors. This store and forward activity continues until all devices receive the update. This process is called *reliable flooding*. Two steps are taken to ensure that this flooding effectively transmits changes without overloading the network with excessive quantities of LSA traffic:

- Each router stores the LSA for a period of time before propagating the information to its neighbors. If, during that time, a new copy of the LSA arrives, the router replaces the stored version. However, if the new copy is outdated, it is discarded.
- To ensure reliability, each link state advertisement must be acknowledged. Multiple acknowledgements can be grouped together into a single acknowledgement packet. If an acknowledgement is not received, the original link state update packet is retransmitted.

Link state advertisements contain five types of information. Together these advertisements provide the necessary information needed to describe the entire OSPF network and any external environments:

- Router LSAs: This type of advertisement describes the state of the router's interfaces (links) within the area. They are generated by every OSPF router. The advertisements are flooded throughout the area.
- Network LSAs: This type of advertisement lists the routers connected to a multi-access network. They are generated by the DR on a multi-access segment. The advertisements are flooded throughout the area.
- Summary LSAs (Type-3 and Type-4): This type of advertisement is generated by an ABR. There are two types of summary link advertisements:
  - Type-3 summary LSAs describe routes to destinations in other areas within the OSPF network (inter-area destinations).
  - Type-4 summary LSAs describe routes to ASBRs. Summary LSAs are used to exchange reachability information between areas. Normally, information is announced into the backbone area. The backbone then injects this information into other areas.
- AS external LSAs: This type of advertisement describes routes to destinations external to the OSPF network. They are generated by an ASBR. The advertisements are flooded throughout all areas in the OSPF network.

Archived

Figure 5-16 illustrates the different types of link state advertisements.

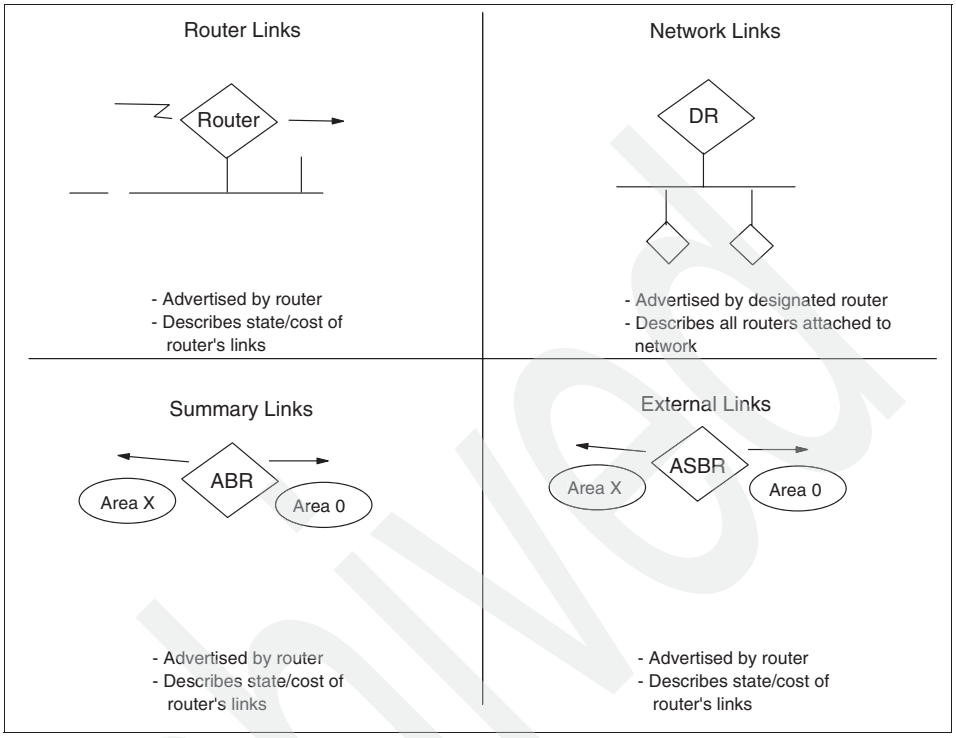


Figure 5-16 OSPF link state advertisements

OSPF packet types

OSPF packets are transmitted in IP datagrams. They are not encapsulated within TCP or UDP packets. The IP header uses protocol identifier 89. OSPF packets are sent with an IP ToS of 0 and an IP precedence of internetwork control. This is used to obtain preferential processing for the packets. We discuss ToS and IP precedence further in “Integrated Services” on page 288.

Wherever possible, OSPF uses multicast facilities to communicate with neighboring devices. In broadcast and point-to-point environments, packets are sent to the reserved multicast address 224.0.0.5. RFC 2328 refers to this as the AllSPFRouters address. In non-broadcast environments, packets are addressed to the neighbor’s specific IP address.

All OSPF packets share the common header shown in Figure 5-17. The header provides general information including area identifier, RID, checksum, and authentication information.

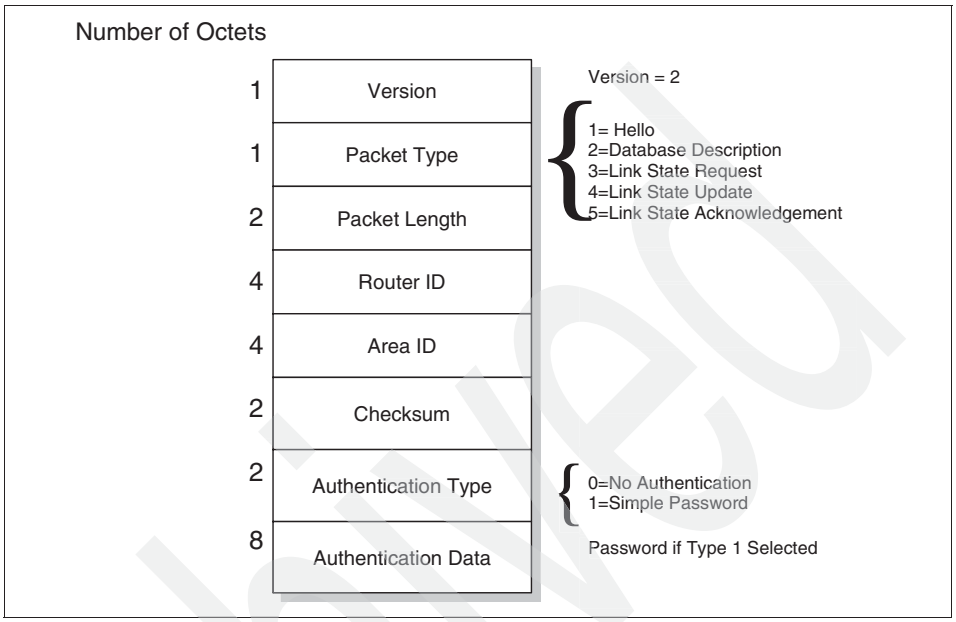


Figure 5-17 OSPF common header

The type field identifies the OSPF packet as one of five possible types:

Hello This packet type discovers and maintains neighbor relationships.

Database description This packet type describes the set of LSAs contained in the router's link state database.

Link state request This packet type requests a more current instance of an LSA from a neighbor.

Link state update This packet type provides a more current instance of an LSA to a neighbor.

Link state acknowledgement This packet type acknowledges receipt of a newly received LSA.

We describe the use of these packets in the next section.

5.6.2 Neighbor communication

OSPF is responsible for determining the optimum set of paths through a network. To accomplish this, each router exchanges LSAs with other routers in the network. The OSPF protocol defines a number of activities to accomplish this information exchange:

- Discovering neighbors
- Electing a designated router
- Establishing adjacencies and synchronizing databases

The five OSPF packet types are used to support these information exchanges.

Discovering neighbors: The OSPF Hello protocol

The Hello protocol discovers and maintains relationships with neighboring routers. Hello packets are periodically sent out to each router interface. The packet contains the RID of other routers whose hello packets have already been received over the interface.

When a device sees its own RID in the hello packet generated by another router, these devices establish a neighbor relationship.

The hello packet also contains the router priority, DR identifier, and BDR identifier. These parameters are used to elect the DR on multi-access networks.

Electing a designated router

All multi-access networks must have a DR. A BDR can also be selected. The backup ensures there is no extended loss of routing capability if the DR fails.

The DR and BDR are selected using information contained in hello packets. The device with the highest OSPF router priority on a segment becomes the DR for that segment. The same process is repeated to select the BDR. In case of a tie, the router with the highest RID is selected. A router declared the DR is ineligible to become the BDR.

After elected, the DR and BDR proceed to establish adjacencies with all routers on the multi-access segment.

Establishing adjacencies and synchronizing databases

Neighboring routers are considered adjacent when they have synchronized their link state databases. A router does not develop an adjacency with every neighboring device. On multi-access networks, adjacencies are formed only with the DR and BDR. This is a two step process.

Step 1: Database exchange process

The first phase of database synchronization is the database exchange process. This occurs immediately after two neighbors attempt to establish an adjacency. The process consists of an exchange of database description packets. The packets contain a list of the LSAs stored in the local database.

During the database exchange process, the routers form a master/subordinate relationship. The master is the first to transmit. Each packet is identified by a sequence number. Using this sequence number, the subordinate acknowledges each database description packet from the master. The subordinate also includes its own set of link state headers in the acknowledgements.

Step 2: Database loading

During the database exchange process, each router notes the link state headers for which the neighbor has a more current instance (all advertisements are time stamped). After the process is complete, each router requests the more current information from the neighbor. This request is made with a link state request packet.

When a router receives a link state request, it must reply with a set of link state update packets providing the requested LSA. Each transmitted LSA is acknowledged by the receiver. This process is similar to the reliable flooding procedure used to transmit topology changes throughout the network.

Every LSA contains an age field indicating the time in seconds since the origin of the advertisement. The age continues to increase after the LSA is installed in the topology database. It also increases during each hop of the flooding process. When the maximum age is reached, the LSA is no longer used to determining routing information and is discarded from the link state database. This age is also used to distinguish between two otherwise identical copies of an advertisement.

5.6.3 OSPF neighbor state machine

The OSPF specification defines a set of neighbor states and the events that can cause a neighbor to transition from one state to another. A state machine is used to describe these transitions:

- Down: This is the initial state. It indicates that no recent information has been received from any device on the segment.
- Attempt: This state is used on non-broadcast networks. It indicates that a neighbor appears to be inactive. Attempts continue to reestablish contact.

- Init: Communication with the neighbor has started, but bidirectional communication has not been established. Specifically, a hello packet was received from the neighbor, but the local router was not listed in the neighbor's hello packet.
- 2-way: Bidirectional communication between the two routers has been established. Adjacencies can be formed. Neighbors are eligible to be elected as designated routers.
- ExStart: The neighbors are starting to form an adjacency.
- Exchange: The two neighbors are exchanging their topology databases.
- Loading: The two neighbors are synchronizing their topology databases.
- Full: The two neighbors are fully adjacent and their databases are synchronized.

Network events cause a neighbor's OSPF state to change. For example, when a router receives a hello packet from a neighboring device, the OSPF neighbor state changes from Down to Init. When bidirectional communication has been established, the neighbor state changes from Init to 2-Way. RFC 2328 contains a complete description of the events causing a state change.

OSPF virtual links and transit areas

Virtual links are used when a network does not support the standard OSPF network topology. This topology defines a backbone area that directly connects to each additional OSPF area. The virtual link addresses two conditions:

- It can logically connect the backbone area when it is not contiguous.
- It can connect an area to the backbone when a direct connection does not exist.

A virtual link is established between two ABRs sharing a common non-backbone area. The link is treated as a point-to-point link. The common area is known as a *transit area*. Figure 5-18 on page 208 illustrates the interaction between virtual links and transit areas when used to connect an area to the backbone.

Archived

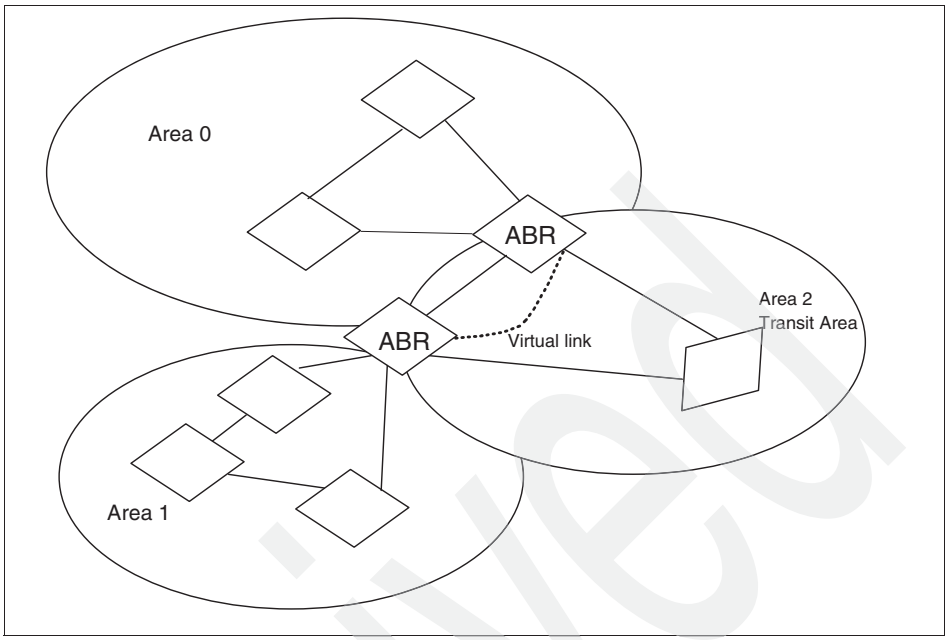


Figure 5-18 OSPF virtual link and transit areas

This diagram shows that area 1 does not have a direct connection to the backbone. Area 2 can be used as a transit area to provide this connection. A virtual link is established between the two ABRs located in area 2. Establishing this virtual link logically extends the backbone area to connect to area 1.

A virtual link is used only to transmit routing information. It does not carry regular traffic between the remote area and the backbone. This traffic, in addition to the virtual link traffic, is routed using the standard intra-area routing within the transit area.

5.6.4 OSPF route redistribution

Route redistribution is the process of introducing external routes into an OSPF network. These routes can be either static routes or routes learned through another routing protocol. They are advertised into the OSPF network by an ASBR. These routes become OSPF external routes. The ASBR advertises these routes by flooding OSPF AS external LSAs throughout the entire OSPF network.

The routes describe an end-to-end path consisting of two portions:

- External portion: This is the portion of the path external to the OSPF network. When these routes are distributed into OSPF, the ASBR assigns an initial cost. This cost represents the *external cost* associated with traversing the external portion of the path.
- Internal portion: This is the portion of the path internal to the OSPF network. Costs for this portion of the network are calculated using standard OSPF algorithms.

OSPF differentiates between two types of external routes. They differ in the way the cost of the route is calculated. The ASBR is configured to redistribute the route as:

- External type 1: The total cost of the route is the sum of the external cost and any internal OSPF costs.
- External type 2: The total cost of the route is always the external cost. This ignores any internal OSPF costs required to reach the ASBR.

Figure 5-19 illustrates an example of the types of OSPF external routes.

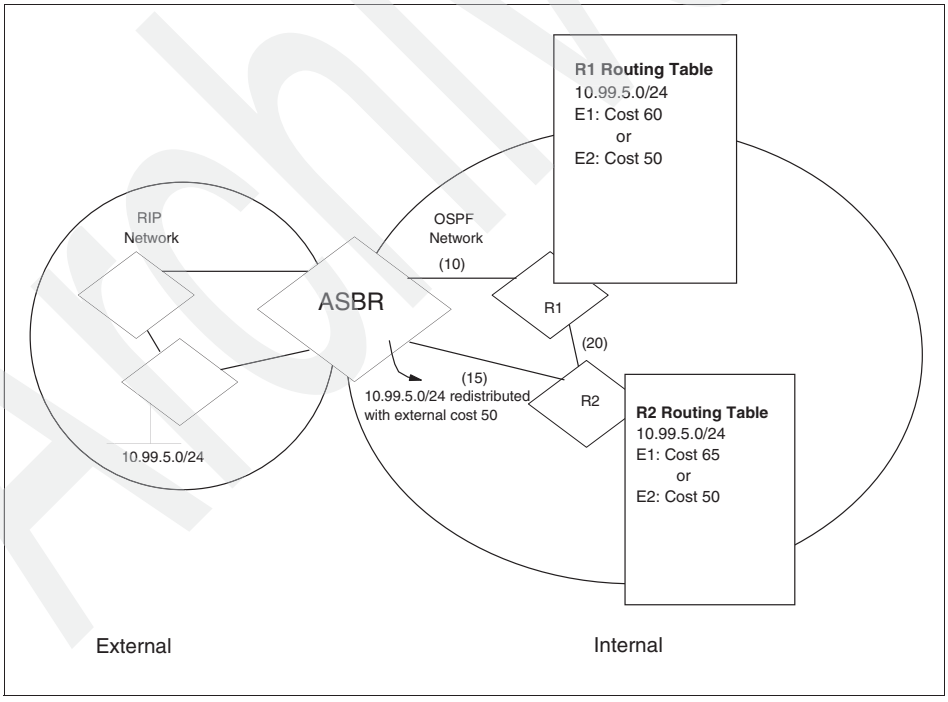


Figure 5-19 OSPF route redistribution

Archived

In this example, the ASBR is redistributing the 10.99.5.0/24 route into the OSPF network. This subnet is located within the RIP network. The route is announced into OSPF with an external cost of 50. This represents the cost for the portion of the path traversing the RIP network:

- If the ASBR redistributed the route as an E1 route, R1 will contain an external route to this subnet with a cost of 60 (50 + 10). R2 will have an external route with a cost of 65 (50 + 15).
- If the ASBR redistributed the route as an E2 route, both R1 and R2 will contain an external route to this subnet with a cost of 50. Any costs associated with traversing segments within the OSPF network are not included in the total cost to reach the destination.

5.6.5 OSPF stub areas

OSPF allows certain areas to be defined as a stub area. A stub area is created when the ABR connecting to a stub area excludes AS external LSAs from being flooded into the area. This is done to reduce the size of the link state database maintained within the stub area routers. Because there are no specific routes to external networks, routing to these destinations is based on a default route generated by the ABR. The link state databases maintained within the stub area contain only the default route and the routes from within the OSPF environment (for example, intra-area and inter-area routes).

Because a stub area does not allow external LSAs, a stub area cannot contain an ASBR. No external routes can be generated from within the stub area.

Stub areas can be deployed when there is a single exit point connecting the area to the backbone. An area with multiple exit points can also be a stub area. However, there is no guarantee that packets exiting the area will follow an optimal path. This is due to the fact that each ABR generates a default route. There is no ability to associate traffic with a specific default routes.

All routers within the area must be configured as stub routers. This configuration is verified through the exchange of hello packets.

Not-so-stubby areas

An extension to the stub area concept is the *not-so-stubby area* (NSSA). This alternative is documented in RFC 3101. An NSSA is similar to a stub area in that the ABR servicing the NSSA does not flood any external routes into the NSSA. The only routes flooded into the NSSA are the default route and any other routes from within the OSPF environment (for example, intra-area and inter-area).

However, unlike a stub area, an ASBR can be located within an NSSA. This ASBR can generate external routes. Therefore, the link state databases

maintained within the NSSA contain the default route, routes from within the OSPF environment (for example, intra-area and inter-area routes), and the external routes generated by the ASBR within the area.

The ABR servicing the NSSA floods the external routes from within the NSSA throughout the rest of the OSPF network.

5.6.6 OSPF route summarization

Route summarization is the process of consolidating multiple contiguous routing entries into a single advertisement. This reduces the size of the link state database and the IP routing table. In an OSPF network, summarization is performed at a border router. There are two types of summarization:

- Inter-area route summarization: Inter-area summarization is performed by the ABR for an area. It is used to summarize route advertisements originating within the area. The summarized route is announcement into the backbone. The backbone receives the aggregated route and announces the summary into other areas.
- External route summarization: This type of summarization applies specifically to external routes injected into OSPF. This is performed by the ASBR distributing the routes into the OSPF network. Figure 5-20 illustrates an example of OSPF route summarization.

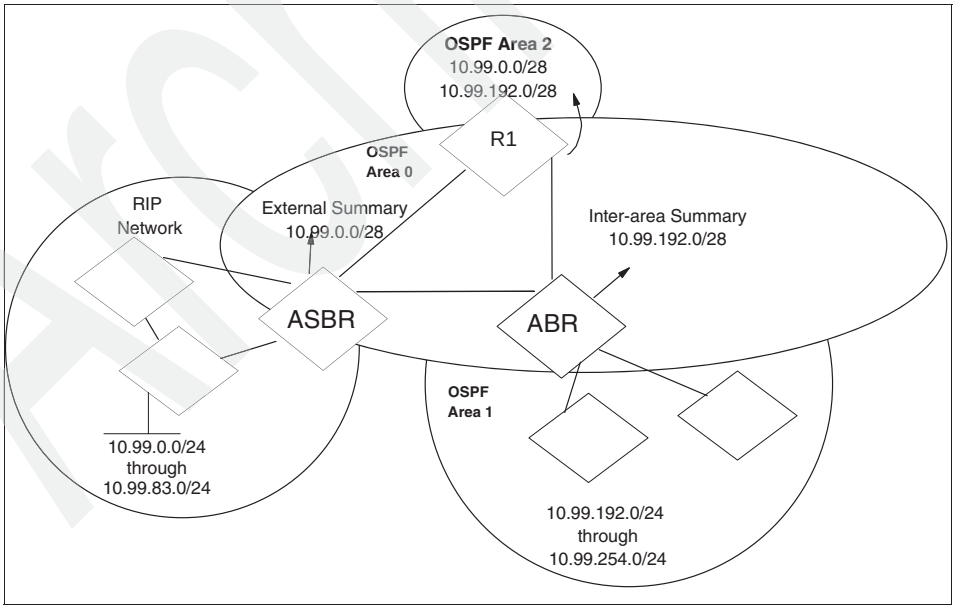


Figure 5-20 OSPF route summarization

In this figure, the ASBR is advertising a single summary route for the 64 subnetworks located in the RIP environment. This single summary route is flooded throughout the entire OSPF network. In addition, the ABR is generating a single summary route for the 64 subnetworks located in area 1. This summary route is flooded through area 0 and area 2. Depending of the configuration of the ASBR, the inter-area summary route can also be redistributed into the RIP network.

5.7 Enhanced Interior Gateway Routing Protocol (EIGRP)

The Enhanced Interior Gateway Routing Protocol (EIGRP) is categorized as a hybrid routing protocol. Similar to a distance vector algorithm, EIGRP uses metrics to determine network paths. However, like a link state protocol, topology updates in an EIGRP environment are event driven.

EIGRP, as the name implies, is an interior gateway protocol designed for use within an AS. In properly designed networks, EIGRP has the potential for improved scalability and faster convergence over standard distance vector algorithms. EIGRP is also better positioned to support complex, highly redundant networks.

EIGRP is a proprietary protocol developed by Cisco Systems, Inc. At the time of this writing, it is not an IETF standard protocol.

5.7.1 Features of EIGRP

EIGRP has several capabilities. Some of these capabilities are also available in distance vector or link state algorithms.

- EIGRP maintains a list of alternate routes that can be used if a preferred path fails. When the path fails, the new route is immediately installed in the IP routing table. No route recomputation is performed.
- EIGRP allows partial routing updates. When EIGRP discovers a neighboring router, each device exchanges their entire routing table. After the initial information exchange, only routing table changes are propagated. There is no periodic rebroadcasting of the entire routing table.
- EIGRP uses a low amount of bandwidth. During normal network operations, only hello packets are transmitted through a stable network.
- EIGRP supports supernetting (CIDR) and variable length subnet masks (VLSM). This enables the network administrator to efficiently allocate IP address resources.

Archived

- EIGRP supports the ability to summarize routing annoucements. This limits the advertisement of unnecessary subnet information.
- EIGRP can provide network layer routing for multiple protocols such as AppleTalk, IPX, and IP networks.
- EIGRP supports the simultaneous use of multiple unequal cost paths to a destination. Each route is installed in the IP routing table. EIGRP also intelligently load balances traffic over the multiple paths.
- EIGRP uses a topology table to install routes into the IP routing table. The topology table lists all destination networks currently advertised by neighboring routers. The table contains all the information needed to build a set of distances and vectors to each destination.
- EIGRP maintains a table to track the state of each adjacent neighbor. This is called a neighbor table.
- EIGRP can guarantee the ordered delivery of packets to a neighbor. However, not all types of packets must be reliably transmitted. For example, in a network that supports multicasting, there is no need to send individual, acknowledged hello packets to each neighbor. To provide efficient operation, reliability is provided only when needed. This improves convergence time in networks containing varying speed connections.

Neighbor discovery and recovery

EIGRP can dynamically learn about other routers on directly attached networks. This is similar to the Hello protocol used for neighbor discovery in an OSPF environment.

Devices in an EIGRP network exchange hello packets to verify each neighbor is operational. Like OSPF, the frequency used to exchange packets is based on the network type. Packets are exchanged at a five second interval on high bandwidth links (for example, LAN segments). Otherwise, hello packets on lower bandwidth connections are exchanged every 60 seconds.

Also like OSPF, EIGRP uses a hold timer to remove inactive neighbors. This timer indicates the amount of time that a device will continue to consider a neighbor active without receiving a hello packet from the neighbor.

EIGRP routing algorithm

EIGRP does not rely on periodic updates to converge on the topology. Instead, it builds a topology table containing each of its neighbor's advertisements. Unlike a distance vector protocol, this data is not discarded.

EIGRP processes the information in the topology table to determine the best paths to each destination network. EIGRP implements an algorithm known as Diffusing Update ALgorithm (DUAL).

Route recomputation

For a specific destination, the successor is the neighbor router currently used for packet forwarding. This device has the least-cost path to the destination and is guaranteed not to be participating in a routing loop. A feasible successor assumes forwarding responsibility when the current successor router fails. The set of feasible successors represent the devices that can become a successor without requiring a route recomputation or introducing routing loops.

A route recomputation occurs when there is no known feasible successor to the destination. The successor is the neighbor router currently used for packet forwarding. The process starts with a router sending a multicast query packet to determine if any neighbor is aware of a feasible successor to the destination. A neighbor replies if it has an feasible successor.

If the neighbor does not have a feasible successor, the neighbor can return a query indicating it also is performing a route recomputation. When the link to a neighbor fails, all routes that used that neighbor as the only feasible successor require a route recomputation.

5.7.2 EIGRP packet types

EIGRP uses five types of packets to establish neighbor relationships and advertise routing information:

- Hello/acknowledgement: These packets are used for neighbor discovery. They are multicast advertised on each network segment. Unicast responses to the hello packet are returned. A hello packet without any data is considered an acknowledgement.
- Updates: These packets are used to convey reachability information for each destination. When a new neighbor is discovered, unicast update packets are exchanged to allow each neighbor to build their topology table. Other types of advertisements (for example, metric changes) use multicast packets. Update packets are always transmitted reliably.
- Queries and replies: These packets are exchanged when a destination enters an active state. A multicast query packet is sent to determine if any neighbor contains a feasible successor to the destination. Unicast reply packets are sent to indicate that the neighbor does not need to go into an active state because a feasible successor has been identified. Query and reply packets are transmitted reliably.

- Request: These packets are used to obtain specific information from a neighbor. These packets are used in route server applications.

5.8 Exterior Gateway Protocol (EGP)

EGP is an exterior gateway protocol of historical merit. It was one of the first protocols developed for communication between autonomous systems. It is described in RFC 904.

EGP assumes the network contains a single backbone and a single path exists between any two autonomous systems. Due to this limitation, the current use of EGP is minimal. In practice, EGP has been replaced by BGP.

EGP is based on periodic polling using a hello/I-hear-you message exchange. These are used to monitor neighbor reachability and solicit update responses.

The gateway connecting to an AS is permitted to advertise only those destination networks reachable within the local AS. It does not advertise reachability information about its EGP neighbors outside the AS.

5.9 Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) is an exterior gateway protocol. It was originally developed to provide a loop-free method of exchanging routing information between autonomous systems. BGP has since evolved to support aggregation and summarization of routing information.

BGP is an IETF draft standard protocol described in RFC 4271. The version described in this RFC is BGP Version 4. Following standard convention, this document uses the term BGP when referencing BGP Version 4.

Archived

5.9.1 BGP concepts and terminology

BGP uses specific terminology to describe the operation of the protocol. Figure 5-21 illustrates this terminology.

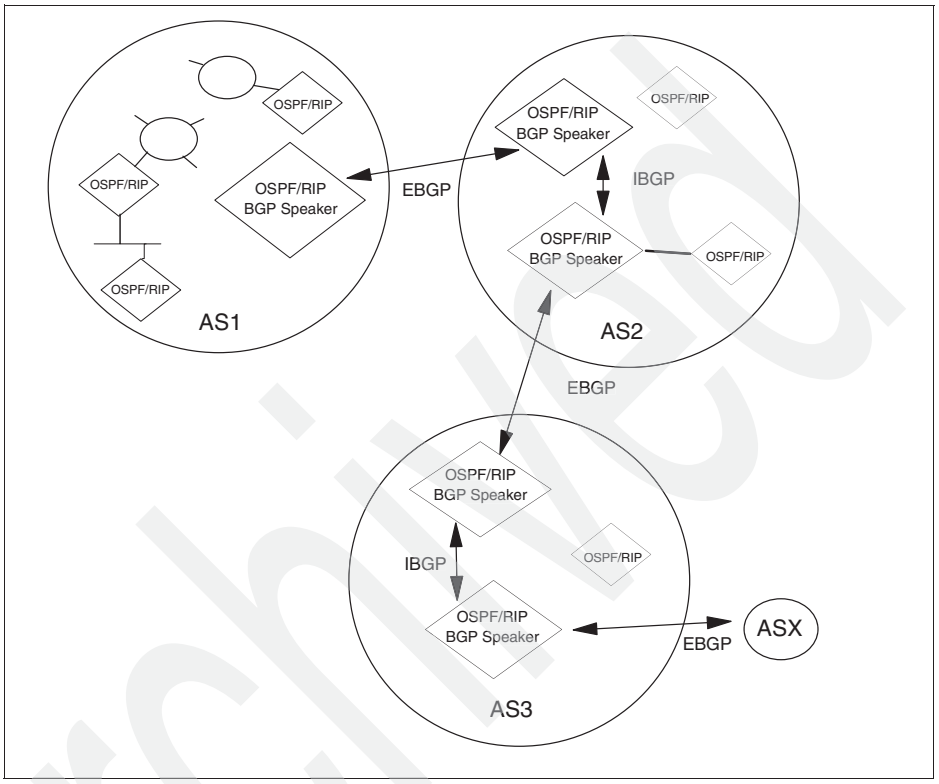


Figure 5-21 Components of a BGP network

BGP uses the following terms:

- BGP speaker: A router configured to support BGP.
- BGP neighbors (peers): A pair of BGP speakers that exchange routing information. There are two types of BGP neighbors:
  - Internal (IBGP) neighbor: A pair of BGP speakers within the same AS.
  - External (EBGP) neighbor: A pair of BGP neighbors, each in a different AS. These neighbors typically share a directly connected network.
- BGP session: A TCP session connecting two BGP neighbors. The session is used to exchange routing information. The neighbors monitor the state of the session by sending keepalive messages.[1]

- Traffic type: BGP defines two types of traffic:
  - Local: Traffic local to an AS either originates or terminates within the AS. Either the source or the destination IP address resides in the AS.
  - Transit: Any traffic that is not local traffic is transit traffic. One of the goals of BGP is to minimize the amount of transit traffic.
- AS type: BGP defines three types of autonomous systems:
  - Stub: A stub AS has a single connection to one other AS. A stub AS carries only local traffic.
  - Multihomed: A multihomed AS has connections to two or more autonomous systems. However, a multihomed AS has been configured so that it does not forward transit traffic.
  - Transit: A transit AS has connections to two or more autonomous systems and carries both local and transit traffic. The AS can impose policy restrictions on the types of transit traffic that will be forwarded.

  Depending on the configuration of the BGP devices within AS 2 in Figure 5-21 on page 216, this autonomous system can be either a multihomed AS or a transit AS.
- AS number: A 16-bit number uniquely identifying an AS.
- AS path: A list of AS numbers describing a route through the network. A BGP neighbor communicates paths to its peers.
- Routing policy: A set of rules constraining the flow of data packets through the network. Routing policies are not defined in the BGP protocol. Rather, they are used to configure a BGP device. For example, a BGP device can be configured so that:
  - A multihomed AS can refuse to act as a transit AS. This is accomplished by advertising only those networks contained within the AS.
  - A multihomed AS can perform transit AS routing for a restricted set of adjacent autonomous systems. It does this by tailoring the routing advertisements sent to EBGP peers.
  - An AS can optimize traffic to use a specific AS path for certain categories of traffic.
- Network layer reachability information (NLRI): NLRI is used by BGP to advertise routes. It consists of a set of networks represented by the tuple <length,prefix>. For example, the tuple <14,220.24.106.0> represents the CIDR route 220.24.106.0/14.

[1] This keepalive message is implemented in the application layer. It is independent of the keepalive message available in many TCP implementations.

- Routes and paths: A route associates a destination with a collection of attributes describing the path to the destination. The destination is specified in NRLI format. The path is reported as a collection of path attributes. This information is advertised in UPDATE messages. For additional information describing the UPDATE message, see 5.9.3, “Protocol description” on page 220.

5.9.2 IBGP and EBGP communication

BGP does not replace the IGP operating within an AS. Instead, it cooperates with the IGP to establish communication between autonomous systems. BGP within an AS is used to advertise the local IGP routes. These routes are advertised to BGP peers in other autonomous systems. Figure 5-22 on page 219 illustrates the communication that occurs between BGP peers. This example shows four autonomous systems. AS 2, AS 3, and AS 4 each have an EBGP connection to AS 1. A full mesh of IBGP sessions exists between BGP devices within AS 1.

Network 10.0.0.0/8 is located within AS 3. Using BGP, the existence of this network is advertised to the rest of the environment:

- R4 in AS 3 uses its EBGP connection to announce the network to AS 1.
- R1 in AS 1 uses its IBGP connections to announce the network to R2 and R3.
- R2 in AS 1 uses its EBGP session to announce the network into AS 2. R3 in AS 1 uses its EBGP session 5 to announce the network into AS 4.

Archived

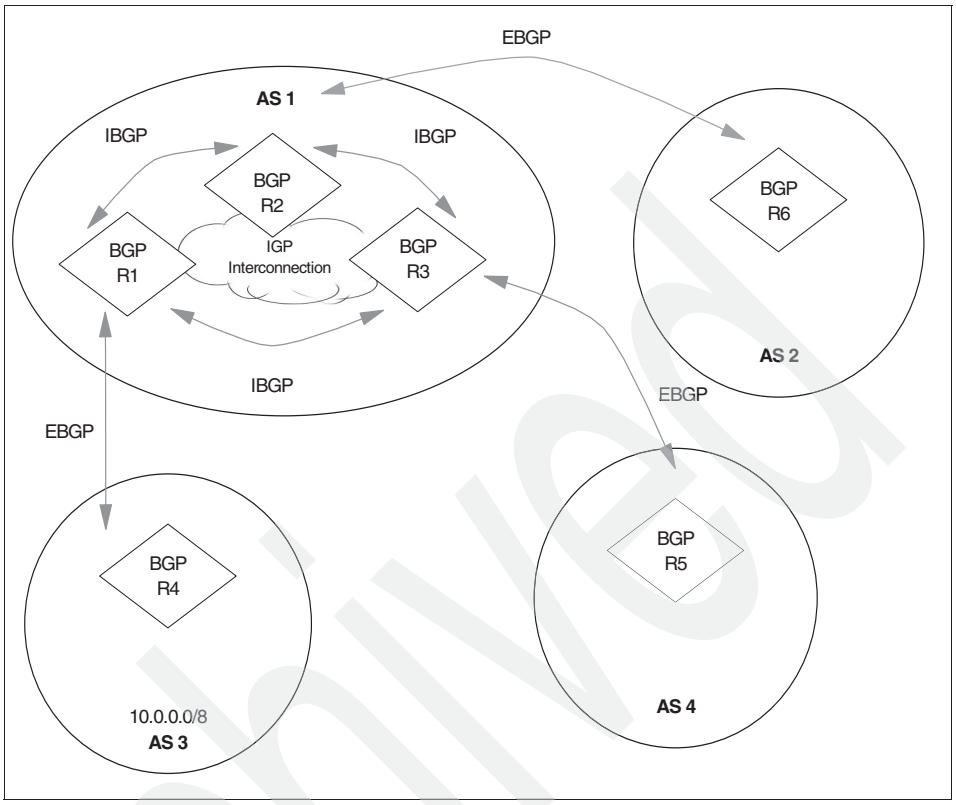


Figure 5-22 EBGP and IBGP communication

Several additional operational issues are shown in Figure 5-22:

- Role of BGP and the IGP: The diagram shows that while BGP alone carries information between autonomous systems, both BGP and the IGP are used to carry information through an AS.
- Establishing the TCP session between peers: Before establishing a BGP session, a device verifies that routing information is available to reach the peer:
  - EBGP peers: EBGP peers typically share a directly connected network. The routing information needed to exchange BGP packets between these peers is trivial.
  - IBGP peers: IBGP peers can be located anywhere within the AS. They do not need to be directly connected. BGP relies on the IGP to locate a peer. Packet forwarding between IBGP peers uses IGP-learned routes.

Archived

- Full mesh of BGP sessions within an AS: IBGP speakers assume a full mesh of BGP sessions have been established between peers in the same AS. In Figure 5-22 on page 219, all three BGP peers in AS 1 are interconnected with BGP sessions.

When a BGP speaker receives a route update from an IBGP peer, the receiving speaker uses EBGP to propagate the update to external peers. Because the receiving speaker assumes a full mesh of IBGP sessions have been established, it does not propagate the update to other IBGP peers.

For example, assume that there was no IBGP session between R1 and R3 in Figure_82. R1 receives the update about 10.0.0.0/8 from AS 3. R1 forwards the update to its BGP peers, namely R2. R2 receives the IBGP update and forwards it to its EBGP peers, namely R6. No update is sent to R3. If R3 needs to receive this information, R1 and R3 must be configured to be BGP peers.

5.9.3 Protocol description

BGP establishes a reliable TCP connection between peers. Sessions are established using TCP port 179. BGP assumes the transport connection will manage fragmentation, retransmission, acknowledgement, and sequencing.

When two speakers initially form a BGP session, they exchange their entire routing table. This routing information contains the complete AS path used to reach each destination. The information avoids the routing loops and counting-to-infinity behavior observed in RIP networks. After the entire table has been exchanged, changes to the table are communicated as incremental updates.

BGP packet types

All BGP packets contain a standard header. The header specifies the BGP packet type. The valid BGP packet types include:

- OPEN[2]: This message type establishes a BGP session between two peer nodes.
- UPDATE: This message type transfers routing information between GP peers.
- NOTIFICATION: This message is sent when an error condition is detected.
- KEEPALIVE: This message determines if peers are reachable.

[2] RFC 1771 uses uppercase to name BGP messages. The same convention is used in this section.

Figure 5-23 shows the flow of these message types between two autonomous systems.

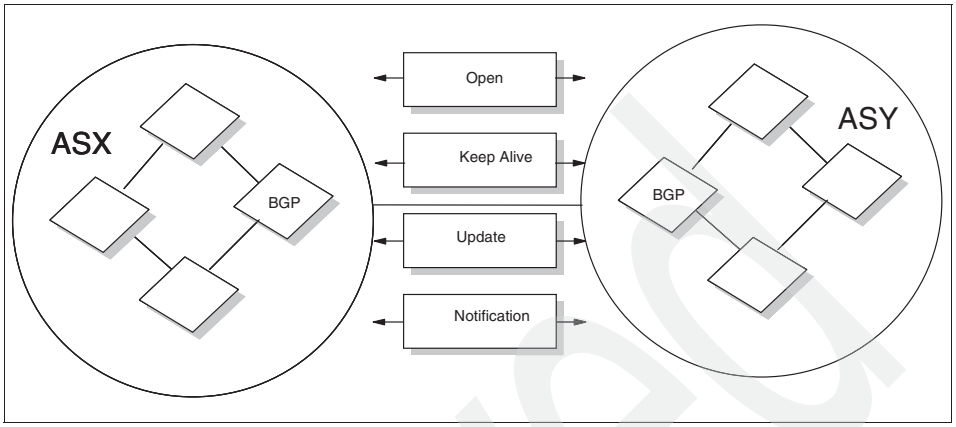


Figure 5-23 BGP message flows between BGP speakers

Opening and confirming a BGP connection

After a TCP session has been established between two peer nodes, each router sends an OPEN message to the neighbor. The OPEN message includes:

- The originating router's AS number and BGP router identifier.
- A suggested value for the hold timer. We discuss the function of this timer in the next section.
- Optional parameters. This information is used to authenticate a peer.

An OPEN message contains support for authenticating the identity of a BGP peer. However, the BGP standard does not specify a specific authorization mechanism. This allows BGP peers to select any supported authorization scheme.

An OPEN message is acknowledged by a KEEPALIVE message. After peer routers have established a BGP connection, they can exchange additional information.

Maintaining the BGP connection

BGP does not use any transport-based keepalive to determine if peers are reachable. Instead, BGP messages are periodically exchanged between peers. If no messages are received from the peer for the duration specified by the hold timer, the originating router assumes that an error has occurred. When this happens, an error notification is sent to the peer and the connection is closed.

RFC 4271 recommends a 90 second hold timer and a 30 second keepalive timer.

Sending reachability information

Reachability information is exchanged between peers in UPDATE messages. BGP does not require a periodic refresh of the entire BGP routing table. Therefore, each BGP speaker must retain a copy of the current BGP routing table used by each peer. This information is maintained for the duration of the connection. After neighbors have performed the initial exchange of complete routing information, only incremental updates to that information are exchanged.

An UPDATE message is used to advertise feasible routes or withdraw infeasible routes. The message can simultaneously advertise a feasible route and withdraw multiple infeasible routes from service. Figure 5-24 depicts the format of an UPDATE message:

- Network layer reachability information (NLRI).
- Path attributes (we discuss path attributes in “Path attributes” on page 223.
- Withdrawn routes.

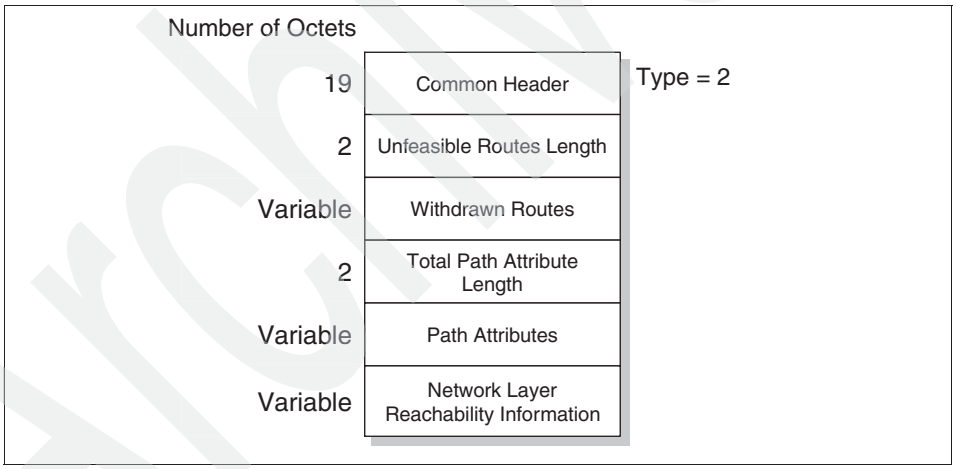


Figure 5-24 BGP UPDATE message

Several path attributes can be used to describe a route.

Withdrawn routes

The unfeasible routes length field indicates the total length of the withdrawn routes field.

Archived

The withdrawn routes field provides a list of IP addresses prefixes that are not feasible or are no longer in service. These addresses need to be withdrawn from the BGP routing table. The withdrawn routes are represented in the same tuple-format as the NLRI.

Notification of error conditions

A BGP device can observe error conditions impacting the connection to a peer. NOTIFICATION messages are sent to the neighbor when these conditions are detected. After the message is sent, the BGP transport connection is closed. This means that all resources for the BGP connection are deallocated. The routing table entries associated with the remote peer are marked as invalid. Finally, other peers are notified that these routes are invalid.

Notification messages include an error code and an error subcode.The error codes provided by BGP include:

- Message header error
- OPEN message error
- UPDATE message error
- Hold timer expired
- Finite state machine error
- Cease

The error subcode further qualifies the specific error. Each error code can have multiple subcodes associated with it.

5.9.4 Path selection

BGP is a path vector protocol. In path vector routing, the path is expressed in terms of the domains (or confederations) traversed so far. The best path is obtained by comparing the number of domains of each feasible route. However, inter-AS routing complicates this process. There are no universally agreed-upon metrics that can be used to evaluate external paths. Each AS has its own set of criteria for path evaluation.

Path attributes

Path attributes are used to describe and evaluate a route. Peers exchange path attributes along with other routing information. When a device advertises a route, it can add or modify the path attributes before advertising the route to a peer. The combination of attributes are used to select the best path.

Archive

Each path attribute is placed into one of four separate categories:

- Well-known mandatory: The attribute must be recognized by all BGP implementations. It must be sent in every UPDATE message.
- Well-known discretionary: The attribute must be recognized by all BGP implementations. However, it is not required to be sent in every UPDATE message.
- Optional transitive: It is not required that every BGP implementation recognize this type of attribute. A path with an unrecognized optional transitive attribute is accepted and simply forwarded to other BGP peers.
- Optional non-transitive: It is not required that every BGP implementation recognize this type of attribute. These attributes can be ignored and not passed along to other BGP peers.

BGP defines seven attribute types to define an advertised route:

- ORIGIN: This attribute defines the origin of the path information. Valid selections are IGP (interior to the AS), EGP, or INCOMPLETE. This is a well-known mandatory attribute.
- AS_PATH: This attribute defines the set of autonomous systems that must be traversed to reach the advertised network. Each BGP device prepends its AS number onto the AS path sequence before sending the routing information to an EBGP peer. Using the sample network depicted in Figure 5-22 on page 219, R4 advertises network 10.0.0.0 with an AS_PATH of 3. When the update traverses AS 1, R2 prepends its own AS number to it. When the routing update reaches R6, the AS_PATH attribute for network 10.0.0.0 is <1 3>. This is a well-known mandatory attribute.
- NEXT_HOP: This attribute defines the IP address of the next hop used to reach the destination. This is a well-known mandatory attribute.

  For routing updates received over EBGP connections, the next hop is typically the IP address of the EBGP neighbor in the remote AS. BGP specifies that this next hop is passed without modification to each IBGP neighbor. As a result, each IBGP neighbor must have a route to reach the neighbor in the remote AS. Figure 5-25 on page 225 illustrates this interaction.

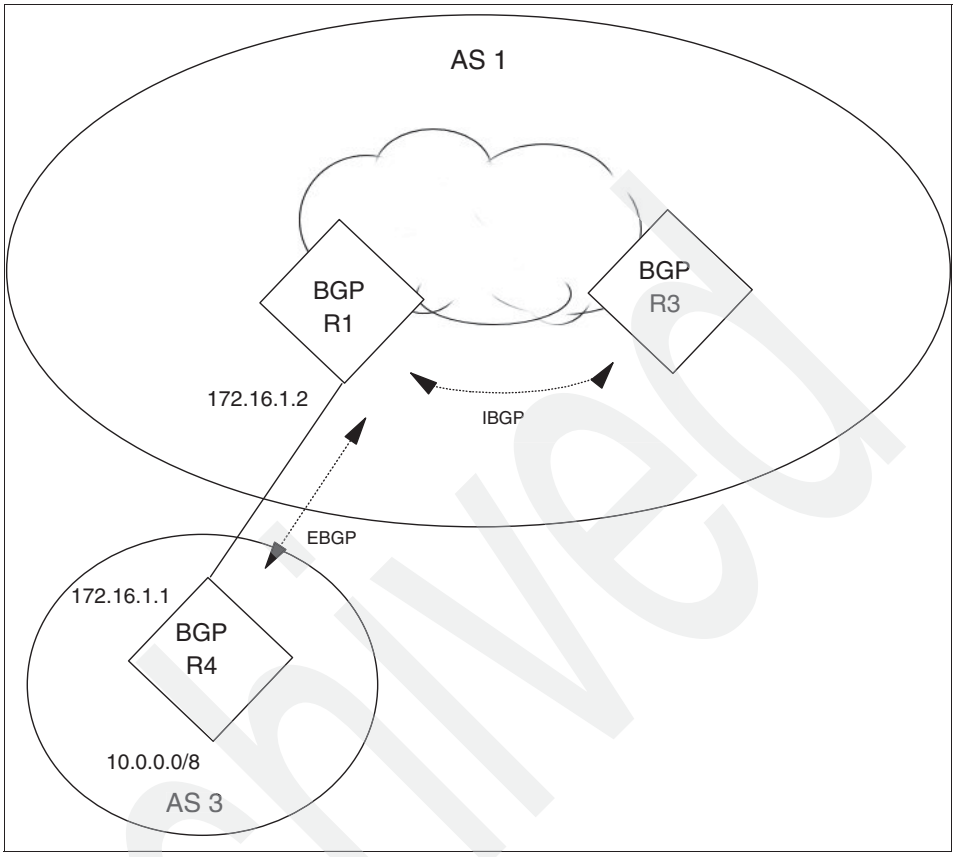


Figure 5-25 NEXT_HOP attribute

In this example, when a routing update for network 10.0.0.0/8 is sent from AS 3, R1 receives the update with the NEXT_HOP attribute set to 172.16.1.1. When this update is forwarded to R3, the next hop address remains 172.16.1.1. R3 must have appropriate routing information to reach this address. Otherwise, R3 will drop packets destined for AS 3 if the next hop is inaccessible.

- MULTI_EXIT_DISC (multi-exit discriminator, MED): This attribute is used to discriminate among multiple exit points to a neighboring AS. If this information is received from an EBGP peer, it is propagated to each IBGP peer. This attribute is not propagated to peers in other autonomous systems. If all other attributes are equal, the exit point with the lowest MED value is preferred. This is an optional non-transitive attribute. MED is discussed further in RFC 4451.

- LOCAL_PREF (local preference): This attribute is used by a BGP speaker to inform other speakers within the AS of the originating speaker's degree of preference for the advertised route. Unlike MED, this attribute is used only within an AS. The value of the local preference is not distributed outside an AS. If all other attributes are equal, the route with the higher degree of preference is preferred. This is a well-known discretionary attribute.
- ATOMIC_AGGREGATE: This attribute is used when a BGP peer receives advertisements for the same destination identified in multiple, non-matching routes (that is, overlapping routes). One route describes a smaller set of destinations (a more specific prefix), other routes describe a larger set of destinations (a less specific prefix). This attribute is used by the BGP speaker to inform peers that it has selected the less specific route without selecting the more specific route. This is a well-known discretionary attribute. A route with this attribute included may actually traverse autonomous systems not listed in the AS_PATH.
- AGGREGATOR: This attribute indicates the last AS number that formed the aggregate route, followed by the IP address of the BGP speaker that formed the aggregate route. For further information about route aggregation, refer to 5.9.6, “BGP aggregation” on page 228. This is an optional transitive attribute.

Decision process

The process to select the best path uses the path attributes describing each route. The attributes are analyzed and a *degree of preference* is assigned. Because there can be multiple paths to a given destination, the route selection process determines the degree of preference for each feasible route. The path with the highest degree of preference is selected as the best path. This is the path advertised to each BGP neighbor. Route aggregation can also be performed during this process. Where there are multiple paths to a destination, BGP tracks each individual path. This allows faster convergence to the alternate path when the primary path fails.

5.9.5 BGP synchronization

Figure 5-26 on page 227 shows an example of an AS providing transit service. In this example, AS 1 is used to transport traffic between AS 3 and AS 4. Within AS 1, R2 is not configured for BGP. However, R2 is used for communication between R1 and R3. Traffic between these two BGP nodes physically traverses through R2.

Using the routing update flow described earlier, the 10.0.0.0/8 network is advertised using the EBGP connection between R4 and R1. R1 passes the network advertisement to R3 using its existing IBGP connection. Because R2 is not configured for BGP, it is unaware of any networks in AS 3. A problem occurs

if R3 needs to communicate with a device in AS 3. R3 passes the traffic to R2. However, because R2 does not have any routes to AS 3 networks, the traffic is dropped.

If R3 advertises the 10.0.0.0/8 network to AS 4, the problem continues. If AS 4 needs to communicate with a device in AS 3, the packets are forwarded from R5 to R3. R3 forwards the packets to R2 where they are discarded.

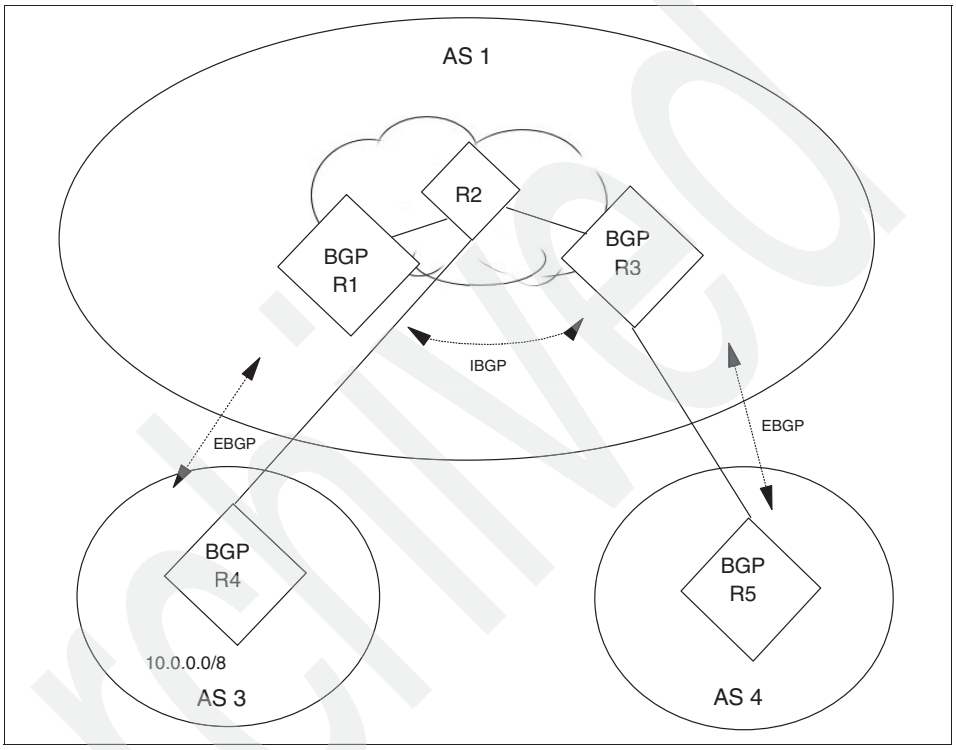


Figure 5-26 BGP synchronization

This situation is addressed by the synchronization rule of BGP. The rule states that a transit AS will not advertise a route before all routers within the AS have learned about the route. In this example, R3 will not advertise the existence of the networks in AS 3 until R2 has built a proper routing table.

There are three methods to implement the synchronization rule:

- Enable BGP on all devices within the transit AS. In this solution, R2 has an IBGP session with both R1 and R3. R2 learns of the 10.0.0.0/8 network at the same time it is advertised to R3. At that time, R3 announces the routes to its peer in AS 4.

- Redistribute the routes into the IGP used within the transit area. In this solution, R1 redistributes the 10.0.0.0/8 network into the IGP within AS 1. R3 learns of the network through two routing protocols: BGP and the IGP. After R3 learns of the network through the IGP, it is certain that other routers within the AS have also learned of the routes. At that time, R3 announces the routes to its peer in AS 4.
- Encapsulate the transit traffic across the AS. In this solution, transit traffic is encapsulated within IP datagrams addressed to the exit gateway. Because this does not require the IGP to carry exterior routing information, no synchronization is required between BGP and the IGP. R3 can immediately announce the routes to its peer in AS 4.

5.9.6 BGP aggregation

The major improvement introduced in BGP Version 4 was support for CIDR and route aggregation. These features allow BGP peers to consolidate multiple contiguous routing entries into a single advertisement. It significantly enhances the scalability of BGP into large internetworking environments. Figure 5-27 on page 229 illustrates these functions.

Archive

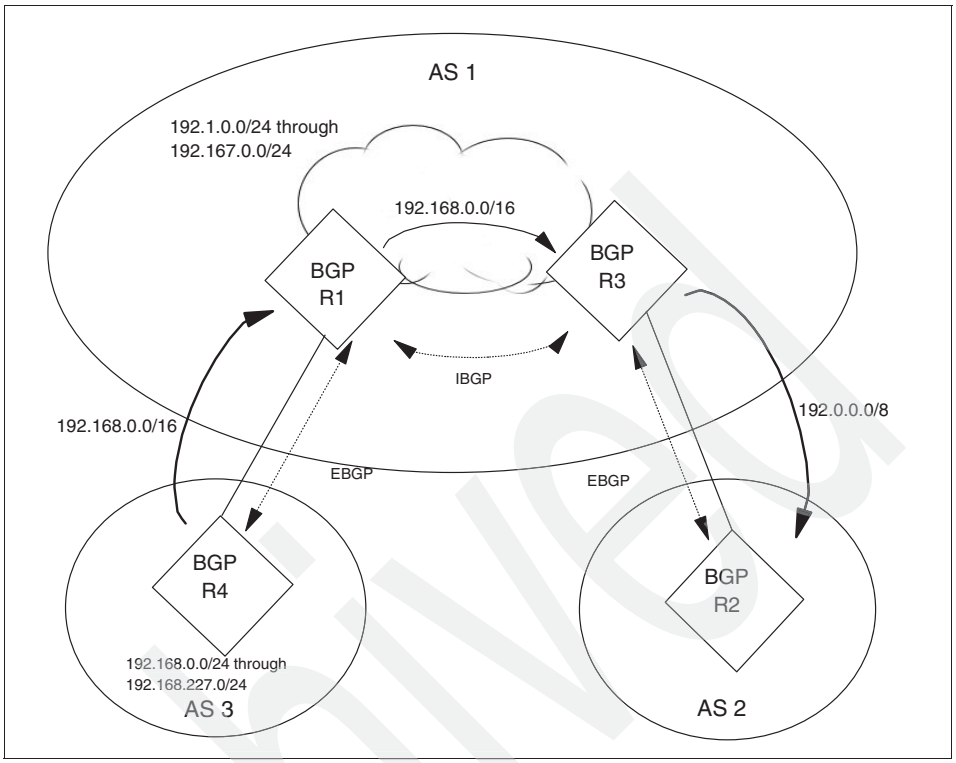


Figure 5-27 BGP route aggregation

This diagrams depicts three autonomous systems interconnected by BGP. In this example, networks 192.168.0.0 through 182.168.227.0 are located within AS 3. To reduce the size of routing announcements, R4 aggregates these individual networks into a single route entry prior to advertising into AS 1. The single entry 192.168.0.0/16 represents a valid CIDR supernet even though it is an illegal Class C network.

BGP aggregate routes contain additional information within the AS_PATH path attribute. When aggregate entries are generated from a set of more specific routes, the AS_PATH attributes of the more specific routes are combined. For example, in Figure 5-27, the aggregate route 192.0.0.0/8 is announced from AS 1 into AS 2. This aggregate represents the set of more specific routes deployed within AS 1 and AS 3. When this aggregate route is sent to AS 2, the AS_PATH attribute consists of <1 3>. This is done to prevent routing information loops. A loop can occur if AS 1 generated an aggregate with an AS_PATH attribute of <1>. If AS 2 had a direct connection to AS 3, the route with the less-specific AS_PATH advertised from AS 1 can generate a loop. This is

because AS 2 does not know this aggregate contains networks located within AS 3.

5.9.7 BGP confederations

BGP requires that all speakers within a single AS have a fully meshed set of IBGP connections. This can be a scaling problem in networks containing a large number of IBGP peers. The use of BGP confederations addresses this problem.

A BGP confederation creates a set of autonomous systems that represent a single AS to peers external to the confederation. This removes the full mesh requirement and reduces management complexity.

Figure 5-28 illustrates the operation of a BGP confederation. In this sample network, AS 1 contains eight BGP speakers. A standard BGP network would require 28 IBGP sessions to fully mesh the speakers.

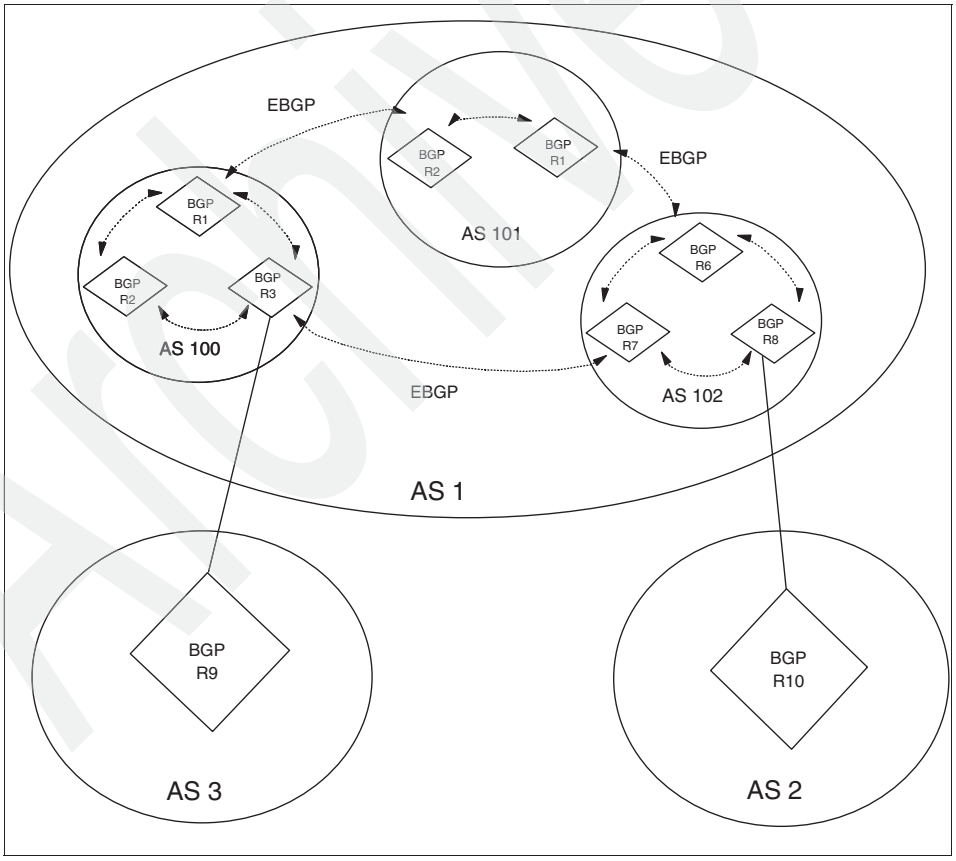


Figure 5-28 BGP confederations

A confederation divides the AS into a set of domains. In this example, AS 1 contains three domains. Devices within a domain have a fully meshed set of IBGP connections. Each domain also has an EBGP connection to other domains within the confederation. In the example network, R1, R2, and R3 have fully meshed IBGP sessions. R1 has an EBGP session within the confederation to R4. R3 has an EBGP session outside the confederation to R9.

Each router in the confederation is assigned a confederation ID. A member of the confederation uses this ID in all communications with devices outside the confederation. In this example, each router is assigned a confederation ID of AS 1.

All communications from AS 1 to AS 2 or AS 3 appear to have originated from the confederation ID of AS 1. Even though communication between domains within a confederation occurs with EBGP, the domains exchange routing updates as though they were connected by IBGP. Specifically, the information contained in the NEXT_HOP, MULTI_EXIT_DESC, and LOCAL_PREF attributes is preserved between domains. The confederation appears to be a single AS to other autonomous systems.

BGP confederations are described in RFC 3065. At the time of this writing, this is a proposed standard. Regardless, BGP confederations have been widely deployed throughout the Internet. Numerous vendors support this feature.

5.9.8 BGP route reflectors

Route reflectors are another solution to address the requirement for a full mesh of IBGP sessions between peers in an AS. As noted previously, when a BGP speaker receives an update from an IBGP peer, the receiving speaker propagates the update only to EBGP peers. The receiving speaker does not forward the update to other IBGP peers.

Route reflectors relax this restriction. BGP speakers are permitted to advertise IBGP learned routes to certain IBGP peers. Figure 5-29 on page 232 depicts an environment using route reflectors. R1 is configured as a route reflector for R2 and R3. R2 and R3 are route reflector clients of R1. No IBGP session is defined between R2 and R3. When R3 receives an EBGP update from AS 3, it is passed to R1 using IBGP. Because R1 is configured as a reflector, R1 forwards the IBGP update to R2.

Figure 5-29 also illustrates the interaction between route reflectors and conventional BGP speakers within an AS.

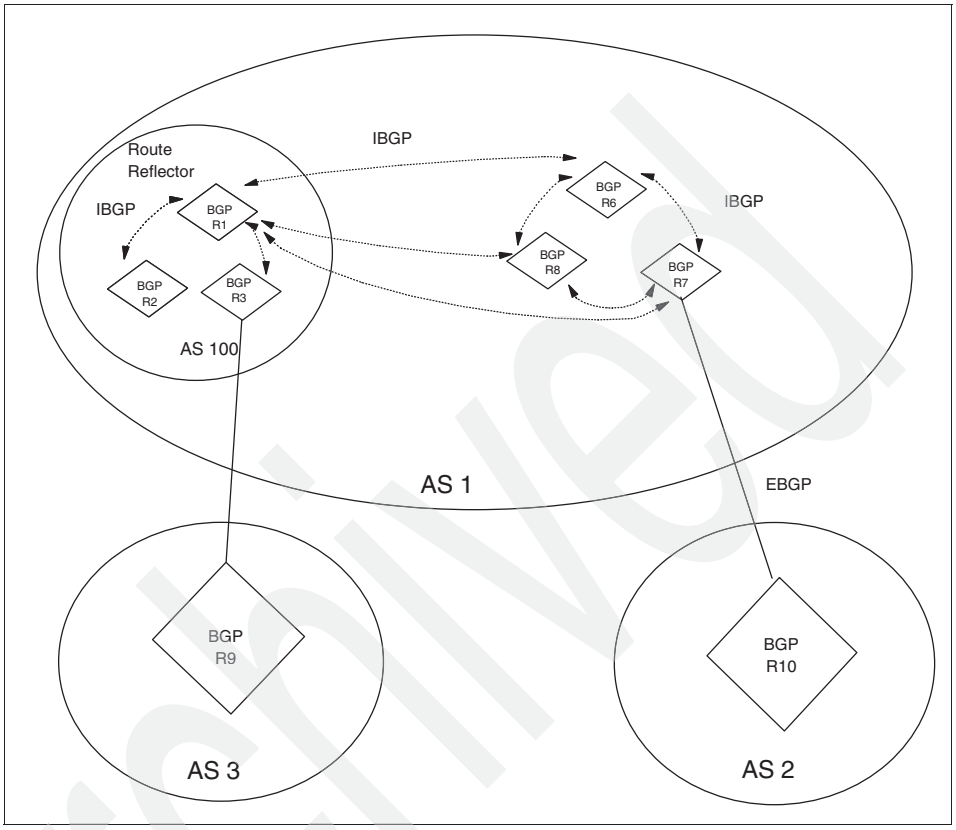


Figure 5-29 BGP route reflector

In Figure 5-29, R1, R2, and R3 are in the route reflector domain. R6, R7, and R8 are conventional BGP speakers containing a full mesh of IBGP peer connections. In addition, each of these speakers is peered with the route reflector. This configuration permits full IBGP communication within AS 1.

Although not shown in Figure 5-29, an AS can contain more than one route reflector. When this occurs, each reflector treats other reflectors as a conventional IBGP peer.

Route reflectors are described in RFC 4456. At the time of this writing, this is a proposed standard.

Archived

5.10 Routing protocol selection

The choice of a routing protocol is a major decision for the network administrator. It has a major impact on overall network performance. The selection depends on network complexity, size, and administrative policies.

The protocol chosen for one type of network might not be appropriate for other types of networks. Each unique environment must be evaluated against a number of fundamental design requirements:

- Scalability to large environments: The potential growth of the network dictates the importance of this requirement. If support is needed for large, highly-redundant networks, consider link state or hybrid algorithms. Distance vector algorithms do not scale into these environments.
- Stability during outages: Distance vector algorithms might introduce network instability during outage periods. The counting to infinity problems (5.3.5, “Convergence and counting to infinity” on page 185) can cause routing loops or other non-optimal routing paths. Link state or hybrid algorithms reduce the potential for these problems.
- Speed of convergence: Triggered updates provide the ability to immediately initiate convergence when a failure is detected. All three types of protocols support this feature. One contributing factor to convergence is the time required to detect a failure. In OSPF and EIGRP networks, a series of hello packets must be missed before convergence begins. In RIP environments, subsequent route advertisements must be missed before convergence in initiated. These detection times increase the time required to restore communication.
- Metrics: Metrics provide the ability to groom appropriate routing paths through the network. Link state algorithms consider bandwidth when calculating routes. EIGRP improves this to include network delay in the route calculation.
- Support for VLSM: The availability of IP address ranges dictates the importance of this requirement. In environments with an constrained supply of addresses, the network administrator must develop an addressing scheme that intelligently overlays the network. VLSM is a major component of this plan. The use of private addresses ranges can also address this concern.
- Vendor interoperability: The types of devices deployed in a network indicate the importance of this requirement. If the network contains equipment from a number of vendors, use standard routing protocols. The IETF has dictated the operating policies for the distance vector and link state algorithms described in this document. Implementing these algorithms avoids any interoperability problems encountered with nonstandard protocols.

- Ease of implementation: Distance vector protocols are the simplest routing protocol to configure and maintain. Because of this, these protocols have the largest implementation base. Limited training is required to perform problem resolution in these environments.

  In small, non-changing environments, static routes are also simple to implement. These definitions change only when sites are added or removed from the network. The administrator must assess the importance of each of these requirements when determining the appropriate routing protocol for an environment.

5.11 Additional functions performed by the router

The main functions performed by a router relate to managing the IP routing table and forwarding data. However, the router should be able to provide information alerting other devices to potential network problems.

This information is provided by the ICMP protocol described in 3.2, “Internet Control Message Protocol (ICMP)” on page 109. The information includes:

- ICMP Destination Unreachable: The destination address specified in the IP packet references an unknown IP network.
- ICMP Redirect: Redirect forwarding of traffic to a more suitable router along the path to the destination.
- ICMP Source Quench: Congestion problems (for example, too many incoming datagrams for the available buffer space) have been encountered in a device along the path to the destination.
- ICMP Time Exceeded: The Time-to-Live field of an IP datagram has reached zero. The packet is not able to be delivered to the final destination.

In addition, each IP router should support the following base ICMP operations and messages:

- Parameter problem: This message is returned to the packet’s source if a problem with the IP header is found. The message indicates the type and location of the problem. The router discards the errored packet.
- Address mask request/reply: A router must implement support for receiving ICMP Address Mask Request messages and responding with ICMP Address Mask Reply messages.
- Timestamp: The router must return a Timestamp Reply to every Timestamp message that is received. It should be designed for minimum variability in delay. To synchronize the clock on the router, the UDP Time Server Protocol or the Network Time Protocol (NTP) can be used.

- Echo request/reply: A router must implement an ICMP Echo server function that receives requests sent to the router and sends corresponding replies. The router can ignore ICMP Echo requests addressed to IP broadcast or IP multicast addresses.

5.12 Routing processes in UNIX-based systems

This chapter focuses on protocols available in standard IP routers. However, several of these protocols are also available in UNIX-based systems.

These protocols are often implemented using one of two processes:

- Routed (pronounced route-D): This is a basic routing process for interior routing. It is supplied with the majority of TCP/IP implementations. It implements the RIP protocol.
- Gated (pronounced gate-D): This is a more sophisticated process allowing for both interior and exterior routing. It can implement a number of protocols including OSPF, RIP-2, and BGP-4.

5.13 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 904 – Exterior Gateway Protocol formal specification (April 1984)
- RFC 1058 – Routing Information Protocol (June 1988)
- RFC 1322 – A Unified Approach to Inter-Domain Routing (May 1992)
- RFC 1812 – Requirements for IP Version 4 Routers. (June 1995)
- RFC 2080 – RIPng for IPv6 (January 1997)
- RFC 2328 – OSPF Version 2 (April 1998)
- RFC 2453 – RIP Version 2 (November 1998)
- RFC 3065 – Autonomous System Confederations for BGP (February 2001)
- RFC 3101 – The OSPF Not-So-Stubby Area (NSSA) Option (January 2003)
- RFC 4271 – A Border Gateway Protocol 4 (BGP-4) (January 2006)
- RFC 4451 – BGP MULTI_EXIT_DISC (MED) Considerations (March 2006)
- RFC 4456 – BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP) (April 2006)

Archived

Archived

6

IP multicast

In early IP networks, a packet could be sent to either a single device (unicast) or to all devices (broadcast). A single transmission destined for a group of devices was not possible. However, during the past few years, a new set of applications has emerged. These applications use multicast transmissions to enable efficient communication between groups of devices. Data is transmitted to a single multicast IP address and received by any device that needs to obtain the transmission.

This chapter describes the interoperation between IP multicasting, Internet Group Management Protocol (IGMP), and multicast routing protocols.

© Copyright IBM Corp. 1989-2006. All rights reserved.

6.1 Multicast addressing

Multicast devices use Class D IP addresses to communicate. These addresses are contained in the range encompassing 224.0.0.0 through 239.255.255.255. For each multicast address, there exists a set of zero or more hosts that listen for packets transmitted to the address. This set of devices is called a host group. A host that sends packets to a specific group does not need to be a member of the group. The host might not even know the current members in the group. There are two types of host groups:

- Permanent: Applications that are part of this type of group have an IP address permanently assigned by the IANA. Membership in this type of host group is not permanent; a host can join or leave the group as required. A permanent group continues to exist even if it has no members. The list of IP addresses assigned to permanent host groups is included in RFC 3232. These reserved addresses include:
  - 224.0.0.0: Reserved base address
  - 224.0.0.1: All systems on this subnet
  - 224.0.0.2: All routers on this subnet
  - 224.0.0.9: All RIP2 routers

  Other address examples include those reserved for OSPF (refer to 5.6, “Open Shortest Path First (OSPF)” on page 196). They include:
  - 224.0.0.5: All OSPF routers
  - 224.0.0.6: OSPF designated routers

  Additionally, IGMPv3 (defined in RFC 3376), reserves the following address:
  - 224.0.0.22: All IGMPv3-capable multicast routers

  An application can use DNS to obtain the IP address assigned to a permanent host group (refer to 12.1, “Domain Name System (DNS)” on page 426) using the domain mcast.net. It can determine the permanent group from an address by using a pointer query (refer to 12.1.6, “Mapping IP addresses to domain names: Pointer queries” on page 430) in the domain 224.in-addr.arpa.
- Transient: Any group that is not permanent is transient. The group is available for dynamic assignment as needed. Transient groups cease to exist when the number of members drops to zero.

6.1.1 Multicasting on a single physical network

This process is straightforward. The sending process specifies a destination IP multicast address. The device driver converts this IP address to the

corresponding Ethernet address and sends the packet to the destination. The destination process informs its network device drivers that it wants to receive datagrams destined for a given multicast address. The device driver enables reception of packets for that address.

In contrast to standard IP unicast traffic forwarding, the mapping between the IP multicast destination address and the data-link address is not done with ARP (see 3.4, “Address Resolution Protocol (ARP)” on page 119). Instead, a static mapping has been defined. In an Ethernet network, multicasting is supported if the high-order octet of the data-link address is 0x'01'. The IANA has reserved the range 0x’01005E000000' through 0x'01005E7FFFFF' for multicast addresses. This range provides 23 usable bits. The 32-bit multicast IP address is mapped to an Ethernet address by placing the low-order 23 bits of the Class D address into the low-order 23 bits of the IANA reserved address block. Figure 6-1 shows the mapping of a multicast IP address to the corresponding Ethernet address.

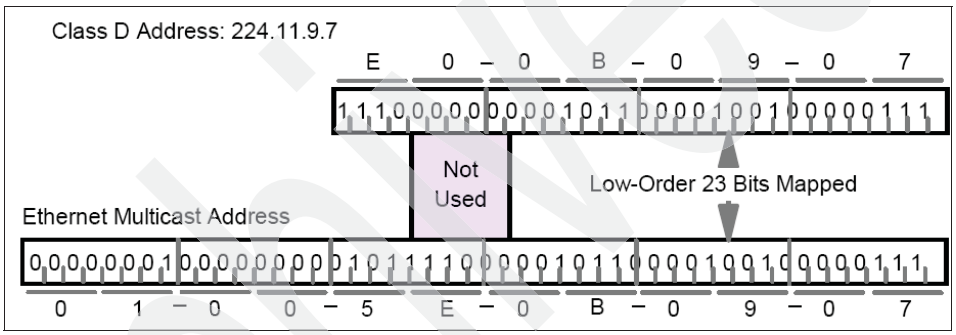


Figure 6-1 Mapping of Class D IP addresses to Ethernet addresses

Because the high-order five bits of the IP multicast group are ignored, 32 different multicast groups are mapped to the same Ethernet address. Because of this non-unique mapping, filtering by the device driver is required. This is done by checking the destination address in the IP header before passing the packet to the IP layer. This ensures the receiving process does not receive spurious datagrams. There are two additional reasons why filtering might be needed:

- Some LAN adapters are limited to a finite number of concurrent multicast addresses. When this limit is exceeded, they receive all multicast packets.
- The filters in some LAN adapters use a hash table value rather than the entire multicast address. If two addresses with the same hash value are used at the same time, the filter might pass excess packets.

Despite this requirement for software filtering, multicast transmissions still cause less inefficiencies for hosts not participating in a specific session. In particular, hosts that are not participating in a host group are not listening for the multicast

address. In this situation, multicast packets are filtered by lower-layer network interface hardware.

6.1.2 Multicasting between network segments

Multicast traffic is not limited to a single physical network. However, there are inherent dangers when multicasting between networks. If the environment contains multiple routers, specific precautions must be taken to ensure multicast packets do not continuously loop through the network. It is simple to create a multicast routing loop. To address this, multicast routing protocols have been developed to deliver packets while simultaneously avoiding routing loops and excess transmissions.

There are two requirements to multicast data across multiple networks:

- Determining multicast participants: A mechanism for determining if a multicast datagram needs to be forwarded on a specific network. This mechanism is defined in RFC 3376, Internet Group Management Protocol (IGMP), Version 3.
- Determining multicast scope: A mechanism for determining the scope of a transmission. Unlike unicast addresses, multicast addresses can extend through the entire Internet.

  The TTL field in a multicast datagram can be used to determine the scope of a transmission. Like other datagrams, each multicast datagram has a Time To Live (TTL) field. The value contained in this field is decremented at each hop. When a host or multicast router receives a datagram, packet processing depends on both the TTL value and the destination IP address:

  - TTL = 0: A multicast datagram received with a TTL value of zero is restricted to the source host.
  - TTL = 1: A multicast datagram with a TTL value of one reaches all hosts on the subnet that are members of the group. Multicast routers decrement the value to zero. However unlike unicast datagrams, no ICMP Time Exceeded error message is returned to the source host. Datagram expiration is a standard occurrence in multicast environments.
  - TTL = 2 (or more): A multicast datagram with this TTL value reaches all hosts on the subnet that are members of the group. The action performed by multicast routers depends on the specific group address:
    - 224.0.0.0 - 224.0.0.255: This range of addresses is intended for single-hop multicast applications. Multicast routers will not forward datagrams with destination addresses in this range.

Even though multicast routers will not forward datagrams within this address range, a host must still report membership in a group within this range. The report is used to inform other hosts on the subnet that the reporting host is a member of the group.

- Other: Datagrams with any other valid Class D destination address are forwarded as normal by the multicast router. The TTL value is decremented by one at each hop.

  This allows a host to implement an *expanding ring search* to locate the nearest server listening to a specific multicast address. The host sends out a datagram with a TTL value of 1 (same subnet) and waits for a reply. If no reply is received, the host resends the datagram with a TTL value of 2. If no reply is received, the host continues to systematically increment the TTL value until the nearest server is found.

6.2 Internet Group Management Protocol (IGMP)

The Internet Group Management Protocol is used by hosts to join or leave a multicast host group. Group membership information is exchanged between a specific host and the nearest multicast router.

IGMP is best regarded as an extension to ICMP. It occupies the same position in the IP protocol stack.

IGMP functions are integrated directly into IPv6, because all IPv6 hosts are required to support multicasting (refer to 9.3.2, “Multicast Listener Discovery (MLD)” on page 365). In IPv4, multicasting and IGMP support is optional.

6.2.1 IGMP messages

IGMP messages are encapsulated in IP datagrams. To indicate an IGMP packet, the IP header contains a protocol number of 2. For IGMP version 2 (defined by RFC 2236), the IP data field contains the 8-octet IGMP message shown in Figure 6-2.

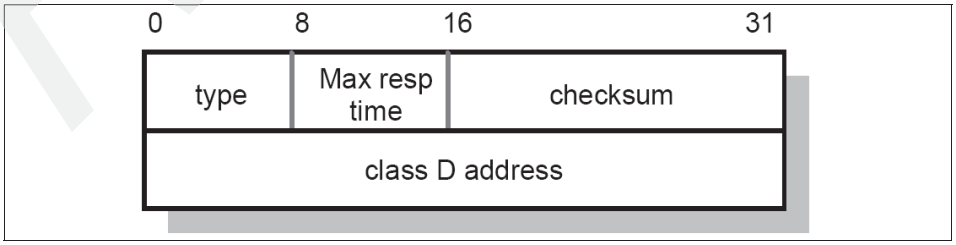


Figure 6-2 IGMP message format

The fields in the IGMP message contain the following information:

- Type: This field specifies the type of IGMP packet:
  - 0x'11': Specifies a membership query packet. This is sent by a multicast router. There are two subtypes of membership query messages:
    - General Query: This is used to learn which groups have members on an attached network.
    - Group-Specific Query: This is used to learn if a particular group has any members on an attached network.
  - 0x'12': Specifies an IGMPv1 membership report packet. This is sent by a multicast host to signal participation in a specific multicast host group.
  - 0x'16': Specifies an IGMPv2 membership report packet.
  - 0x'17': Specifies a leave group packet. This is sent by a multicast host.
- Max resp time: This field is used in membership query messages. It specifies the maximum allowed time a host can wait before sending a corresponding report. Varying this setting allows routers to tune the leave latency. This references the time between the last host leaving a group and the time the routing protocol is notified that there are no more members.
- Checksum: This field contains a 16-bit checksum.
- Class D Address: This field contains a valid multicast group address. It is used in a report packet.

IGMPv3 messages

The IGMPv2 message format has been extended in IGMP version 3, defined in RFC 3376 (which obsoletes RFC 2236). Version 3 allows receivers to subscribe to or exclude a specific set of sources within a multicast group. To accommodate this, The 0x'11' type membership query packet has been altered, and a new IGMP packet type of 0x'22' has been added. However, all IGMPv3 implementations must still support packet types 0x'12', 0x'16', and 0x'17'.

Archived

Figure 6-3 illustrates the expanded version 3 membership query packet.

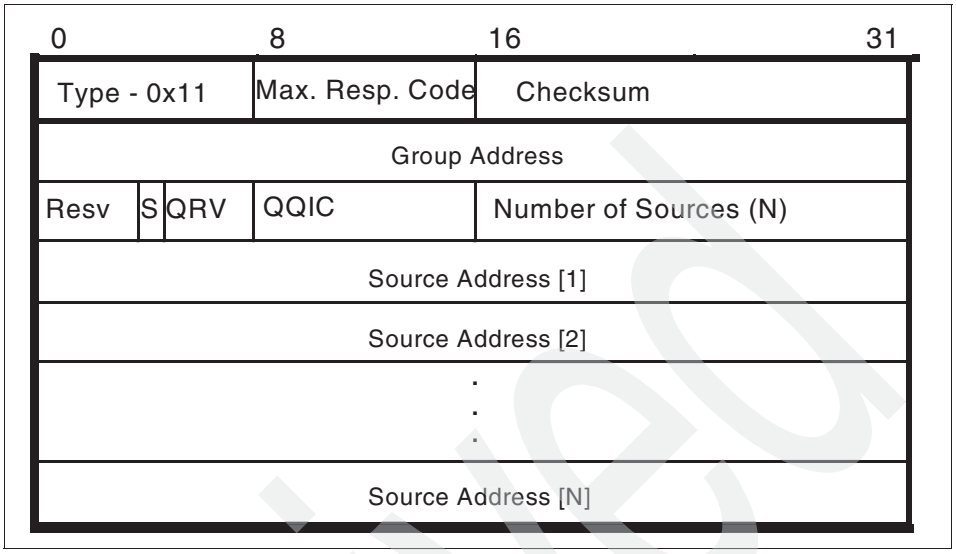


Figure 6-3 The IGMPv3 membership query message

The fields from this message are as follows:

- Type: This remains unchanged and, in the case of the membership query message, has a value of 0x'11'.
- Max resp code: This field is has been changed from the IGMPv2 convention, and distinguishes between a maximum response *code* and a maximum response *time*. The time can be determined from the code as follows:
  - If the maximum response code is less than 128, the value of maximum response code is also the maximum response time.
  - If the maximum response code is greater than 128, the maximum response code represents a floating-point value denoted as follows:
    - Byte 0 = 1
    - Bytes 1-3 = *exp*
    - Bytes 4-7 = *mant*

    Maximum response time = (*mant* OR 0x'10) << (*exp* + 3)

> **Note:** Maximum response time, for both IGMPv2 and IGMPv3, is measured in tenths of a second.

This can be better described as creating a 5-bit string starting with *1* followed by the 4 bits of *mant*, and then shifting this string *exp*+3 to the left. The resulting value is the maximum response time. For example, assume that the value of maximum response code is decimal 178. The bit string representation of this is *10110010*. From this, the fields of the maximum response code are:

- Byte 0 = 1
- *exp* = 011
- *mant* = 0010

The subsequent calculations are (note that the *exp* calculation uses a binary representation of decimal 3):

- (*mant* OR 0x10) = (0010 OR 10000) = 10010
- 10010 << *exp* + 11 = 10010 << 110 = 10010000000
- Binary 10010000000 = Decimal 1152

Therefore, when the maximum response code is decimal 178, the maximum response time is 1152 tenths of a second.

- Checksum: This field contains a 16-bit checksum, and remains unchanged from its version 2 counterpart.
- Group Address: This field contains the Class D address, and remains unchanged from its version 2 counterpart.
- Resv: This field is reserved, and is set to zero on transmission and ignored on reception.
- S Flag: When set to 1, this field indicates that any receiving multicast routers should suppress the normal timer updates normally performed upon hearing a query.
- QRV: This field is the Querier's Robustness Variable. The QRV was added in IGMPv3, and is used in tuning timer values for expected packet loss. The higher the value of the QRV, the more tolerant the environment is for lost packets within a domain. However, increasing the QRV also increases the latency required in detecting a problem. Routers adopt the value in this field from the most recently received query as their own robustness variable. If this value exceeds the limit of 7, it is reset to 0.
- QQIC: This field is the Querier's Query Interval Code. This value, specified in seconds, specifies the query interval used by the originator of this query. The calculations to convert this code into the actual interval time is the same used for the maximum response code.

- Number of Sources (N): This field indicates how many source address are contained within the message. The maximum value for this field is determined by the MTU allowed in the network.
- Source Addresses: This set of fields are a vector of N IP unicast addresses, where the value N corresponds to the Number or Sources (N) field.

Additionally, IGMPv3 adds a new type of 0x'22', which is the IGMPv3 Membership Report. Figure 6-4 illustrates the format for this message.

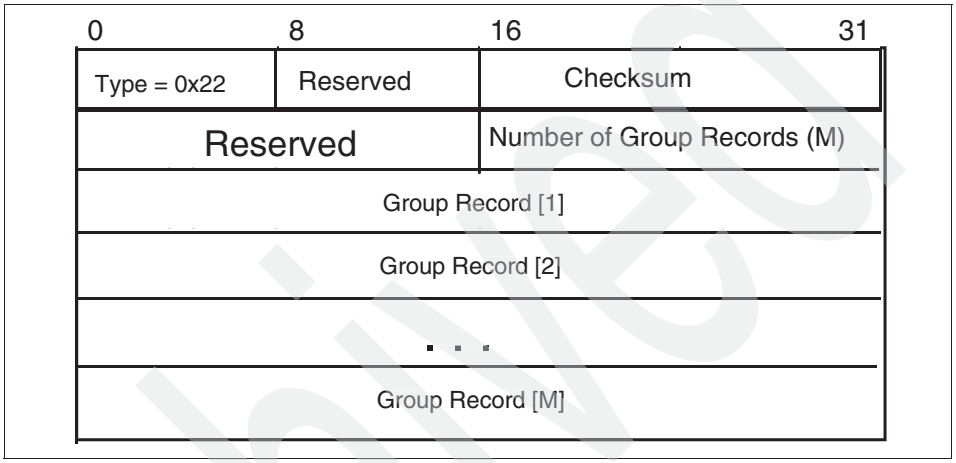


Figure 6-4 The IGMPv3 membership report message

Archived

In Figure 6-4 on page 245, the Type, Checksum, and Number of Group Records fields are equivalent to their IGMPv3 message query counterparts. However, in the membership report message, each Group Record is broken up, as illustrated in Figure 6-5.

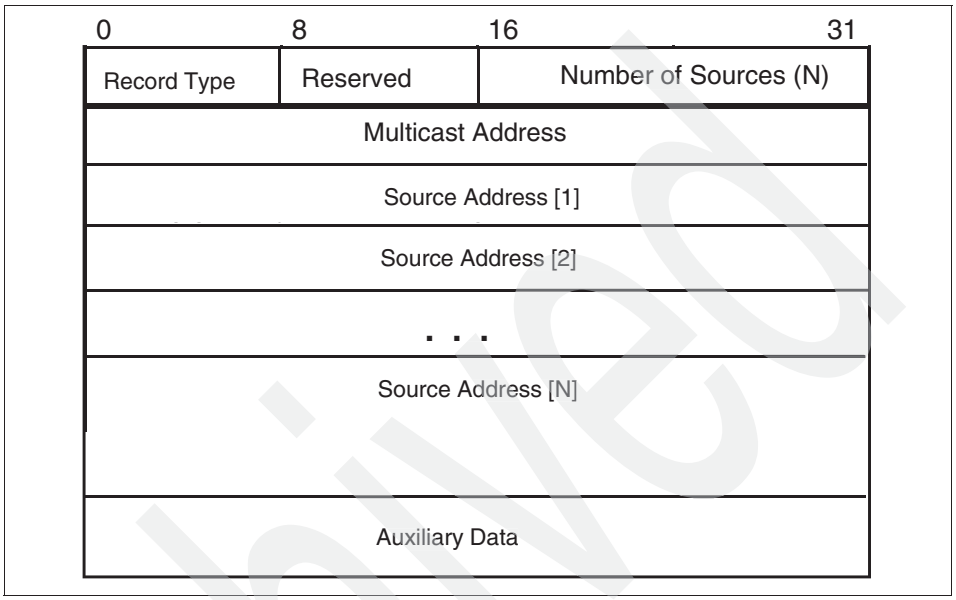


Figure 6-5 IGMPv3 group record

The fields within the group record are described as follows:

- Record Type: This field indicates whether the group record type is a *current-state*, *filter-mode-change*, or *source-list-change* record. The values are as follows:
  - *Current-state* records are sent by a system in response to a query received on an interface, and report the current reception of that interface. There are two group record values that denote a current-state record:
    - MODE_IS_INCLUDE: This indicates that the interface has a filter mode of INCLUDE for the specified multicast addresses.
    - MODE_IS_EXCLUDE: This indicates that the interface has a filter mode of EXCLUDE for the specified multicast addresses.

Archived

– *Filter-mode-change* records are sent by a system whenever an interface's state changes for a particular multicast address. There are two group record values which denote a filter-mode-change record:
  - CHANGE_TO_INCLUDE_MODE: This indicates that the interface has changed to the INCLUDE filter mode for the specified multicast addresses.
  - CHANGE_TO_EXCLUDE_MODE: This indicates that the interface has changed to the EXCLUDE filter mode for the specified multicast addresses.
– *Source-list-change* records are sent by a system whenever an interface wants to alter the list of source addresses without altering its state. There are two group record values that denote a source-list-change record:
  - ALLOW_NEW_SOURCES: This indicates that the interface has changed such that it wants to receive messages from additional sources. If the filter is an INCLUDE filter, the specified multicast addresses will be added. If it is an EXCLUDE filter, the specified multicast addresses will be removed.
  - BLOCK_OLD_SOURCES: This indicates that the interface has changed such that it no longer wants to receives messages from additional sources. If the filter is an INCLUDE filter, the specified multicast addresses will be removed. If it is an EXCLUDE filter, the specified multicast addresses will be added.

We discuss these group record types in greater detail in "IGMPv3 specific host operations" on page 248.

6.2.2 IGMP operation

Both hosts and multicast routers participate in IGMP functions.

Host operations

To receive multicast datagrams, a host must join a host group. When a host is multihomed, it can join groups on one or more of its attached interfaces. If a host joins the same group on multiple interfaces, the multicast messages received by the host can be different. For example, 244.0.0.1 is the group for *all* hosts on this subnet. Messages in this group received through one subnet will always be different from those on another subnet.

Multiple processes on a single host can listen for messages from the same group. When this occurs, the host joins the group once. The host internally tracks each process interested in the group.

To join a group, the host sends an IGMP membership report packet through an attached interface. The report is addressed to the desired multicast group. A host does not need to join the all hosts group (224.0.0.1). Membership in this group is automatic.

IGMPv3 specific host operations

In IGMPv3, hosts specify a list of multicast addresses from which they want to receive messages, or a list of multicast addresses from which they do *not* want to receive messages. Hosts can then later alter these lists to add or remove multicast addresses. This can be achieved using the filter-mode-change and source-list-change records.

Note: If no interface state exists, it is created using the filter-mode-change record.

The use of these records is demonstrated using Table 6-1. In this example, the *current state* indicates what subsets of multicast addresses (A and B) are currently included or excluded. The *desired state* indicates the subsets desired to be included or excluded. The *Records needed to achieve desired state* show the records which can must be sent to achieve this change. Note that the group type records are abbreviated as follows:

- CHANGE_TO_INCLUDE_MODE: To_in
- CHANGE_TO_EXCLUDE_MODE: To_ex
- ALLOW_NEW_SOURCES: Allow
- BLOCK_OLD_SOURCES: Block

Table 6-1 IGMPv3 list changes using group record types

	Current state	Desired state	Records needed to achieve desired state
1	Include ()	Include (A)	To_in (A)
2	Include (A)	Include (B)	Allow (B-A), Block (A-B)
3	Include (B)	Exclude (A)	To_ex (A)
4	Exclude (A)	Exclude (B)	Allow (A-B), Block (B-A)
5	Exclude (B)	Include (A)	To_in (A)

These steps are summarized as follows:

1. No source address lists currently exists. The subset of lists *A* is added by issuing a CHANGE_TO_INCLUDE_MODE specifying the A subset.

2. Subset A is currently included. However, only subset *B* is desired. To do this, first an ALLOW_NEW_SOURCES message is issued to add all of subset B except for those addresses already included in A. This is followed by a BLOCK_OLD_SOURCES message to exclude all of subset A except for those addresses which also belong to B.
3. Now only subset B is included. However, we want to change the filter to EXCLUDE, and to specify only subset A. This is done with one CHANGE_TO_EXCLUDE_MODE specifying subset A.
4. After step 3, only subset A is excluded. Now, we want to exclude only subset B. First issue an ALLOW_NEW_SOURCES message to remove all of subset A from the excluded list except those addresses also in subset B. Then add all of the addresses in subset B to the exclude list except for those also in subset A.
5. Now only subset B is excluded. We want to change the filter to INCLUDE and add A to the list. Use the CHANGE_TO_INCLUDE_MODE specifying subset A.

Multicast router operations

When a host attempts to join a group, multicast routers on the subnet receive the membership report packet and create an entry in their *local group database*. This database tracks the group membership of the router's directly attached networks. Each entry in the database is of the format [group, attached network]. This indicates that the attached network has at least one IP host belonging to the group. Multicast routers listen to all multicast addresses to detect these reports.

The information in the local group database is used to forward multicast datagrams. When the router receives a datagram, it is forwarded out each interface containing hosts belonging to the group.

To verify group membership, multicast routers regularly send an IGMP query message to the all hosts' multicast address. Each host that still wants to be a member of a group sends a reply. RFC 3376 specifies this verification should by default occur every 125 seconds. To avoid bursts of traffic on the subnet, replies to query messages are sent using a random delay. Because routers do not track the number of hosts in each group, any host that hears another device claim membership cancels any pending membership replies. If no hosts claim membership within the specified interval, the multicast router assumes no hosts on that network are members of the group.

IGMP snooping switches

One potential problem when implementing IGMP is the flooding of a network segment with multicast packets, even though there might not be any nodes on that segment that have any interest in receiving the packets. Although the

amount of processing involved in receiving these packets presents no significant cost, the flooding of these packets does have the potential to waste bandwidth.

This problem is documented in RFC 4541, which provides recommendations and suggested rules to allow network switches to “snoop” on IGMP traffic. By doing so, switches can analyze the data contained within the IGMP header and determine if the traffic needs to be forwarded to every segment to which the switch is connected. By doing this, switches can reduce the amount of unnecessary IGMP traffic flooding uninterested networks, thereby reserving the segment’s bandwidth for the traffic with which the hosts on those networks are concerned.

6.3 Multicast delivery tree

IGMP only specifies the communication occurring between receiving hosts and their local multicast router. Routing of packets between multicast routers is managed by a separate routing protocol. We describe these protocols in 6.4, “Multicast forwarding algorithms” on page 252. Figure 6-6 on page 251 shows that multicast routing protocols and IGMP operate in different sections of the *multicast delivery tree*.

Archive

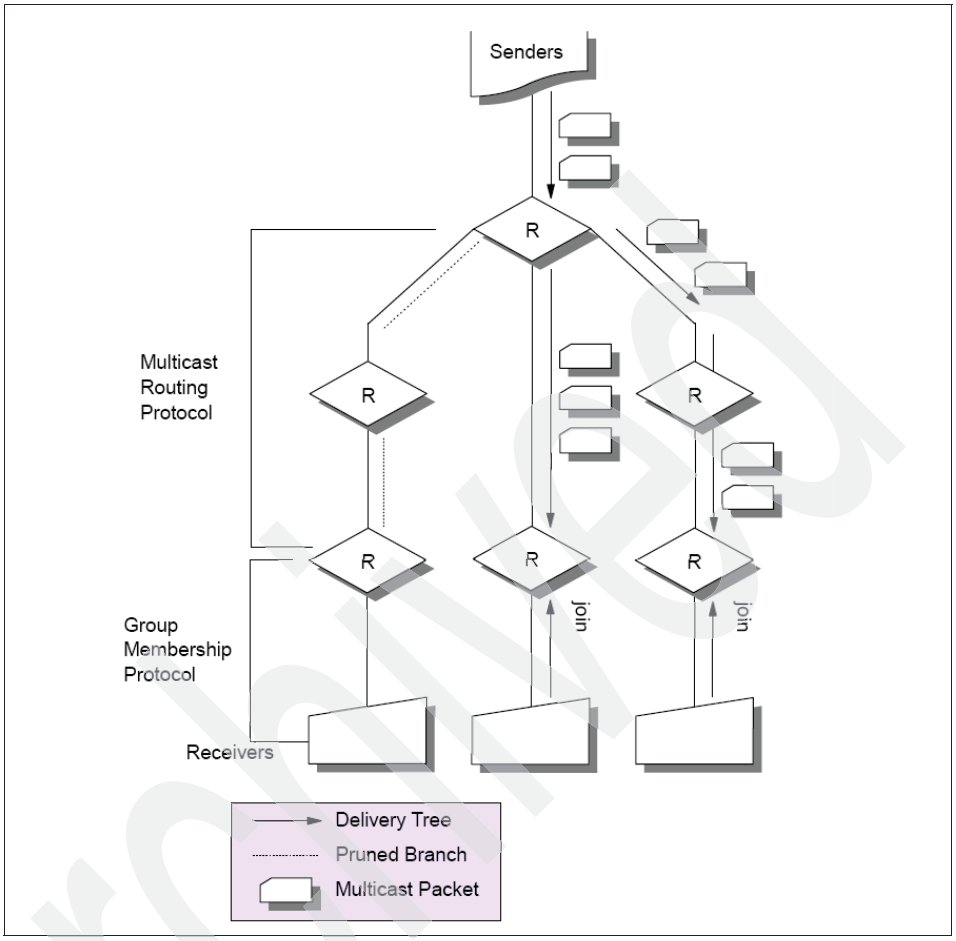


Figure 6-6 Multicast delivery tree

Figure 6-6 shows the tree formed between a multicast sender and the set of receivers. If there are no hosts connected to a multicast router that have joined this specific group, no multicast packets for the group should be delivered to the branch connecting these hosts. The branch are *pruned* from the delivery tree. This action reduces the size of the tree to the minimum number of branches needed to reach every group member. New sections of the tree can be dynamically added as new members join the group. This *grafts* new sections to the delivery tree.

6.4 Multicast forwarding algorithms

Multicast algorithms are used to establish paths through the network. These paths allow multicast traffic to effectively reach all group members. Each algorithm should address the following set of requirements:

- The algorithm must route data only to group members.
- The algorithm must optimize the path from source to destinations.
- The algorithm must maintain loop-free routes.
- The algorithm must provide scalable signaling functions used to create and maintain group membership.
- The algorithm must not concentrate traffic on a subset of links.

Several algorithms have been developed for use in multicast routing protocols. These algorithms have varying levels of success addressing these design requirements. We review two algorithms in the following sections.

6.4.1 Reverse path forwarding algorithm

The reverse path forwarding (RPF) algorithm uses a multicast delivery tree to forward datagrams from the source to each member in the multicast group. As shown in Figure 6-7, packets are replicated only at necessary branches in the delivery tree.

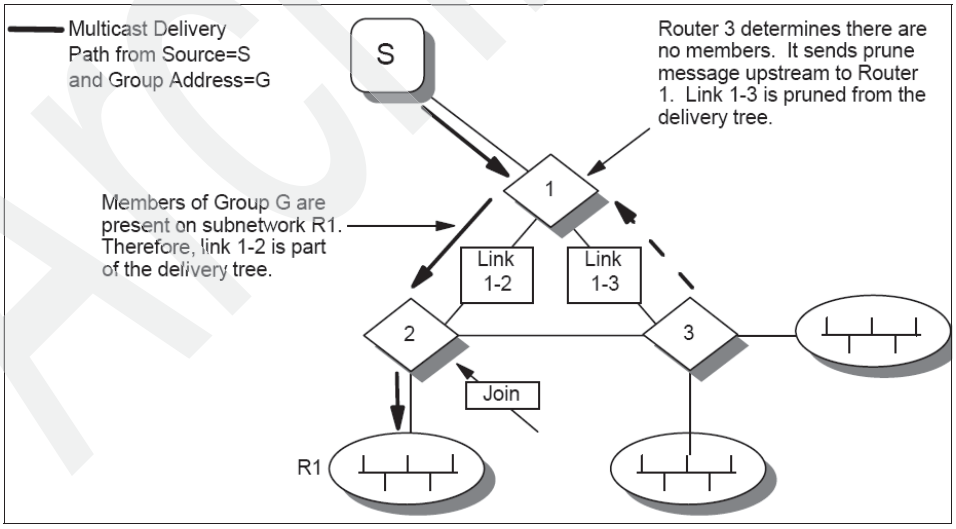


Figure 6-7 Reverse path forwarding (RPF)

To track the membership of individual groups, trees are calculated and updated dynamically.

The algorithm maintains a *reverse path table* used to reach each source. This table maps every known source network to the preferred interface used to reach the source. When forwarding data, if the datagram arrives through the interface used to transmit datagrams back to the source, the datagram is forwarded through every appropriate downstream interface. Otherwise, the datagram arrived through a sub-optimal path and is discarded. Using this process, duplicate packets caused by network loops are filtered.

The use of RPF provides two benefits:

- RPF guarantees the fastest delivery for multicast data. In this configuration, traffic follows the shortest path from the source to each destination.
- A different tree is computed for each source node. Packet delivery is distributed over multiple network links. This results in more efficient use of network resources.

6.4.2 Center-based tree algorithm

The center-based tree (CBT) algorithm describes another method to determine optimum paths between members of a multicast group. The algorithm describes the following steps:

1. A *center point* in the network is chosen. This fixed point represents the center of the multicast group.
2. Each recipient sends a join request directed towards the center point. This is accomplished using an IGMP membership report for that group.
3. The request is processed by all intermediate devices located between the multicast recipient and the center point. If the router receiving the request is already a member of the tree, it marks one more interface as belonging to the group. If this is the first join request, the router forwards the request one step further toward the source.

This procedure builds a delivery tree for each multicast group. The tree is identical for all sources. Each router maintains a single tree for the entire group. This contrasts with the process used in the RPF algorithm. The RPF algorithm builds a tree for each sender in a multicast group.

Because there is no requirement for the source to be a member of the group, multicast packets from a source are forwarded toward the center point until they reach a router belonging to the tree. At this stage, the packets are forwarded using the multicast processing of the center-based tree.

The disadvantage to the center-based tree algorithm is that it might build a suboptimal path for some sources and receivers.

6.4.3 Multicast routing protocols

A number of multicast routing protocols have been developed using these algorithms:

- Distance Vector Multicast Routing Protocol (DVMRP)
- Multicast OSPF (MOSPF)
- Protocol Independent Multicast (PIM)

The remainder of this chapter describes these protocols.

6.5 Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP is an established multicast routing protocol originally defined in RFC 1075. The standard was first implemented as the *mrouted* process available on many UNIX systems. It has since been enhanced to support RPF. DVMRP is an interior gateway protocol. It is used to build per-source per-group multicast delivery trees within an autonomous system (AS).

DVMRP does not route unicast datagrams. Any router that processes both multicast and unicast datagrams must be configured with two separate routing processes. Because separate processes are used, multicast and unicast traffic might not follow the same path through the network.

6.5.1 Protocol overview

DVMRP is described as a *broadcast and prune* multicast routing protocol:

- DVMRP builds per-source broadcast trees based on routing exchanges.
- DVMRP dynamically prunes the per-source broadcast tree to create a multicast delivery tree. DVMRP uses the RPF algorithm to determine the set of downstream interfaces used to forward multicast traffic.

Neighbor discovery

DVMRP routers dynamically discover each neighbor by periodically sending neighbor probe messages on each local interface. These messages are sent to the *all-DVMRP-routers* multicast address (224.0.0.4). Each message contains a list of neighbor DVMRP routers for which neighbor probe messages have been

Archived

received. This allows a DVMRP router to verify it has been seen by each neighbor.

After a router has received a probe message that contains its address in the neighbor list, the pair of routers establish a two-way neighbor adjacency.

Routing table creation

DVMRP computes the set of reverse paths used in the RPF algorithm. To ensure that all DVMRP routers have a consistent view of the path connecting to a source, a routing table is exchanged between each neighbor router. DVMRP implements its own unicast routing protocol. This routing protocol is similar to RIP.

The algorithm is based on hop counts. DVMRP requires a metric to be configured on every interface. Each router advertises the network number, mask, and metric of each interface. The router also advertises routes received from neighbor routers. Like other distance vector protocols, when a route is received, the interface metric is added to the advertised metric. This adjusted metric is used to determine the best upstream path to the source.

DVMRP has one important difference from RIP. RIP manages routing and datagram forwarding to a particular unicast destination. DVMRP manages the return path to the source of a particular multicast datagram.

Dependent downstream routers

In addition to providing a consistent view of paths to source networks, exchanging routing information provides an additional benefit. DVMRP uses this mechanism to notify upstream routers that a specific downstream router requires them to forward multicast traffic.

DVMRP accomplishes this by using the poison reverse technique (refer to “Split horizon with poison reverse” on page 188). If a downstream router selects an upstream router as the next hop to a particular source, routing updates from the downstream router specify a metric of infinity for the source network. When the upstream router receives the advertisement, it adds the downstream router to a list of *dependent downstream routers* for this source. This technique provides the information needed to prune the multicast delivery tree.

Designated forwarder

When two or more multicast routers are connected to a multi-access network, duplicate packets can be forwarded to the network. DVMRP prevents this possibility by electing a designated forwarder for each source.

When the routers exchange their routing table, each learns the peer's metric to reach the source network. The router with the lowest metric is responsible for forwarding data to the shared network. If multiple routers have the same metric, the router with the lowest IP address becomes the designated forwarder for the network.

6.5.2 Building and maintaining multicast delivery trees

As previously mentioned, the RPF algorithm is used to forward multicast datagrams. If a datagram was received on the interface representing the best path to the source, the router forwards the datagram out a set of downstream interfaces. This set contain each downstream interface included in the multicast delivery tree.

Building the multicast delivery tree

A multicast router forwards datagrams to two types of devices: downstream dependent routers and hosts that are members of a particular multicast group. If a multicast router has no dependent downstream neighbors through a specific interface, the network is a *leaf network*. The delivery tree is built using routing information detailing these different types of destinations.

Adding leaf networks

If the downstream interface connects to a leaf network, packets are forwarded only if there are hosts that are members of the specific multicast group. The router obtains this information from the IGMP local group database. If the group address is listed in the database, and the router is the designated forwarder for the source, the interface is included in the multicast delivery tree. If there are no group members, the interface is excluded.

Adding non-leaf networks

Initially, all non-leaf networks are included in the multicast delivery tree. This allows each downstream router to participate in traffic forwarding for each group.

Pruning the multicast delivery tree

Routers connected to leaf networks remove an interface when there are no longer any active members participating in the specific multicast group. When this occurs, multicast packets are no longer forwarded through the interface.

If a router is able to remove all of its downstream interfaces for a specific group, it notifies its upstream neighbor that it no longer needs traffic from that particular source and group pair. This notification is accomplished by sending a prune message to the upstream neighbor. If the upstream neighbor receives prune

messages from each of the dependent downstream routers on an interface, the upstream router can remove this interface from the multicast delivery tree.

If the upstream router is able to prune all of its interfaces from the tree, it sends a prune message to its upstream router. This continues until all unnecessary branches have been removed from the delivery tree.

Maintaining prune information

In order to remove outdated prune information, each prune message contains a prune lifetime timer. This indicates the length of time that the prune will remain in effect. If the interface is still pruned when the timer expires, the interface is reconnected to the multicast delivery tree. If this causes unwanted multicast datagrams to be delivered to a downstream device, the prune mechanism is reinitiated.

Grafting pruned networks

Because IP multicast supports dynamic group membership, hosts can join a multicast group at any time. When this occurs, DVMRP routers use graft messages to reattach the network to the multicast delivery tree. A graft message is sent as a result of receiving a IGMP membership report for a group that has previously been pruned. Separate graft messages are sent to the appropriate upstream neighbor for each source network that has been pruned.

Receipt of a graft message is acknowledged with a graft ACK message. This enables the sender to differentiate between a lost graft packet and an inactive device. If an acknowledgment is not received within the graft timeout period, the request is retransmitted. The purpose of the graft ACK message is to acknowledge the receipt of a graft message. It does not imply any action has been taken as a result of the request. Therefore, all graft request messages are acknowledged even if they do not cause any action to be taken by the receiving router.

Archived

6.5.3 DVMRP tunnels

Some IP routers might not be configured to support native multicast routing. DVMRP provides the ability to tunnel IP multicast datagrams through networks containing non-multicast routers. The datagrams are encapsulated in unicast IP packets and forwarded through the network. This behavior is shown in Figure 6-8.

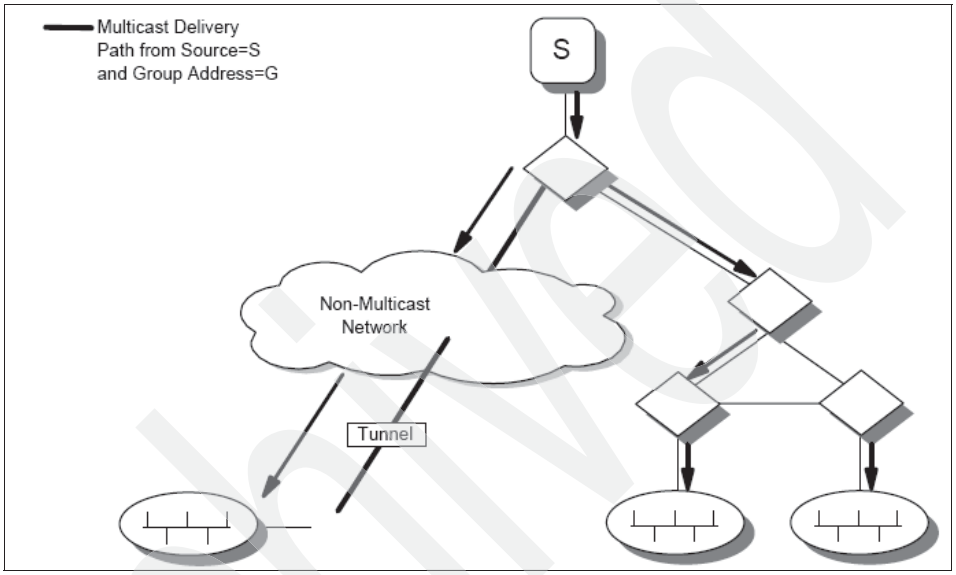


Figure 6-8 DVMRP tunnels

When the packet is received at the remote end of the tunnel, it is decapsulated and forwarded through the subnetwork using standard DVMRP multicast operations.

6.6 Multicast OSPF (MOSPF)

MOSPF is a multicast extension to OSPF Version 2 (refer to 5.6, “Open Shortest Path First (OSPF)” on page 196), defined in RFC 1584. Unlike DVMRP, MOSPF is not a separate multicast routing protocol. It is used in networks that already use OSPF for unicast IP routing. The multicast extensions leverage the existing OSPF topology database to create a *source-rooted shortest path delivery tree*.

MOSPF forwards multicast datagrams using both source and destination address. This contrasts the standard OSPF algorithm, which relies solely on destination address.

6.6.1 Protocol overview

We present an overview of MOSPF in the following sections.

Group-membership LSA

The location of every group member must be communicated to the rest of the environment. This ensures that multicast datagrams are forwarded to each member. OSPF adds a new type of link state advertisement (the group-membership-LSA) to track the location of each group member. These LSAs are stored in the OSPF link state database. This database describes the topology of the AS.

Designated routers

On each network segment, one MOSPF router is selected to be the designated router (DR). This router is responsible for generating periodic IGMP host membership queries. It is also responsible for listening to the IGMP membership reports. Routers ignore any report received on a network where they are not the DR. This ensures that each network segment appears in the local group database of at most one router. It also prevents datagrams from being duplicated as they are delivered to local group members.

Every router floods a group-membership-LSA for each multicast group having at least one entry in the router's local group database. This LSA is flooded throughout the OSPF area.

Shortest-path delivery trees

The path used to forward a multicast datagram is calculated by building a shortest-path delivery tree rooted at the datagram's source (refer to “Shortest-Path First (SPF) algorithm” on page 177). This tree is built from information contained in the link state database. Any branch in the shortest-path delivery tree that does not have a corresponding group-membership-LSA is pruned. These branches do not contain any multicast members for the specific group.

Initially, shortest-path delivery trees are built when the first datagram is received. The results are cached for use by subsequent datagrams having the same source and destination. The tree is recomputed when a link state change occurs or when the cache information times out.

In an MOSPF network, all routers calculate an identical shortest-path delivery tree for a specific multicast datagram. There is a single path between the datagram source and any specific destination. This means that unlike OSPF's treatment of unicast traffic, MOSPF has no provision for equal-cost multipath.

6.6.2 MOSPF and multiple OSPF areas

OSPF allows an AS to be split into areas. Although this has several traffic management benefits, it limits the topology information maintained in each router. A router is only aware of the network topology within the local area. When building shortest-path trees in these environments, the information contained in the link state database is not sufficient to describe the complete path between each source and destination. This can lead to non-optimal path selection.

Within an OSPF area, the area border router (ABR) forwards routing information and data traffic between areas. The corresponding functions in an MOSPF environment are performed by an *inter-area multicast forwarder*. This device forwards group membership information and multicast datagrams between areas. An OSPF ABR can also function as an MOSPF inter-area multicast forwarder.

Because group-membership-LSAs are only flooded within an area, a process to convey membership information between areas is required. To accomplish this, each inter-area multicast forwarder summarizes the attached areas' group membership requirements and forwards this information into the OSPF backbone. This announcement consists of a group-membership-LSA listing each group containing members in the non-backbone area. The advertisement performs the same function as the summary LSAs generated in a standard OSPF area.

However, unlike route summarization in a standard OSPF network, summarization for multicast group membership in MOSPF is asymmetric. Membership information for the non-backbone area is summarized into the backbone. However, this information is not re-advertised into other non-backbone areas.

To forward multicast data traffic between areas, a *wildcard multicast receiver* is used. This is a router to which all multicast traffic, regardless of destination, is forwarded. In non-backbone areas, all inter-area multicast forwarders are wildcard multicast receivers. This ensures that all multicast traffic originating in a non-backbone area is forwarded to a inter-area multicast forwarder. This router sends the multicast datagrams to the backbone area. Because the backbone has complete knowledge of all group membership information, the datagrams are then forwarded to the appropriate group members in other areas.

6.6.3 MOSPF and multiple autonomous systems

An analogous situation to inter-area multicast routing exists when at least one multicast device resides in another AS. In both cases, the shortest path tree describing the complete path from source to destination cannot be built.

In this environment, an ASBR in the MOSPF domain is configured as an *inter-AS multicast forwarder*. This router is also configured with an inter-AS multicast routing protocol. Although the MOSPF standard does not dictate the operations of the inter-AS protocol, it does assume the protocol forwards datagrams using RPF principles. Specifically, MOSPF assumes that a multicast datagram whose source is outside the domain will enter the domain at a point that is advertising (into OSPF) the best route to the source. MOSPF uses this information to calculate the path of the datagram through the domain.

MOSPF designates an inter-AS multicast forwarder as a wildcard multicast receiver. As with inter-area communications, this ensures that the receiver remains on all pruned shortest-path delivery trees. They receive all multicast datagrams, regardless of destination. Because this device has complete knowledge of all group membership outside the AS, datagrams can be forwarded to group members in other autonomous systems.

6.6.4 MOSPF interoperability

Routers configured to support an MOSPF network can be intermixed with non-multicast OSPF routers. Both types of routers interoperate when forwarding unicast data traffic. However, forwarding IP multicast traffic is limited to the MOSPF domain. Unlike DVMRP, MOSPF does not provide the ability to tunnel multicast traffic through non-multicast routers.

6.7 Protocol Independent Multicast (PIM)

The complexity associated with MOSPF lead to the development and deployment of PIM. PIM is another multicast routing protocol. Unlike MOSPF, PIM is independent of any underlying unicast routing protocol. It interoperates with all existing unicast routing protocols.

PIM defines two modes or operation:

- Dense mode (PIM-DM), specified in RFC 3973
- Sparse mode (PIM-SM), specified in RFC 2362

Dense mode and sparse mode refer to the density of group members within an area. In a random sampling, a group is considered dense if the probability of finding at least one group member within the sample is high. This holds even if the sample size is reasonably small. A group is considered sparse if the probability of finding group members within the sample is low.

PIM provides the ability to switch between spare mode and dense mode. It also permits both modes to be used within the same group.

6.7.1 PIM dense mode

The PIM-DM protocol implements the RPF process described in 6.4.1, “Reverse path forwarding algorithm” on page 252. Specifically, when a PIM-DM device receives a packet, it validates the incoming interface with the existing unicast routing table. If the incoming interface reflects the best path back to the source, the router floods the multicast packet. The packet is sent out to every interface that has not been pruned from the multicast delivery tree.

Unlike DVMRP, PIM-DM does not attempt to compute multicast specific routes. Rather, it assumes that the routes in the unicast routing table are symmetric.

Similar to operations in a DVMRP environment, a PIM-DM device initially assumes all downstream interfaces need to receive multicast traffic. The router floods datagrams to all areas of the network. If some areas do not have receivers for the specific multicast group, PIM-DM reactively prunes these branches from the delivery tree. This reactive pruning is done because PIM-DM does not obtain downstream receiver information from the unicast routing table. Figure 6-9 contains an example of a PIM-DM pruning.

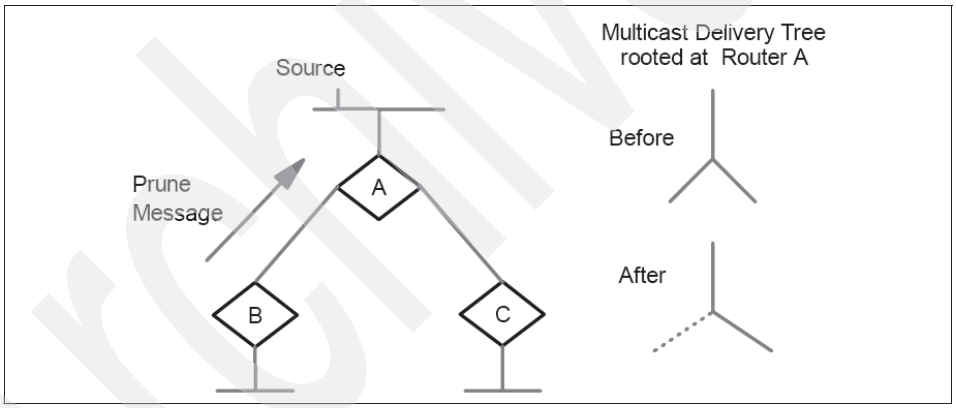


Figure 6-9 PIM-DM flood and prune operation

PIM-DM is relatively simple to implement. The only assumption is that a router is able to retain a list of prune requests.

PIM-DM benefits

Given the flood and prune methodology used in PIM-DM, use this protocol in environments where the majority of hosts within a domain need to receive the multicast data. In these environments, the majority of networks will not be pruned

from the delivery tree. The inefficiencies associated with flooding is minimal. This configuration is also appropriate when:

- Senders and receivers are in close proximity to each other.
- There are few senders and many receivers.
- The volume of multicast traffic is high.
- The stream of multicast traffic is constant.

Unlike DVMRP, PIM-DM does not support tunnels to transmit multicast traffic through non-multicast capable networks. Therefore, the network administrator must ensure that each device connected to the end-to-end path is multicast-enabled.

6.7.2 PIM sparse mode

The PIM-SM protocol uses a variant of the center-based tree algorithm. In a PIM-SM network, a *rendezvous point (RP)* is analogous to the center point described in the algorithm. Specifically, an RP is the location in the network where multicast senders connect to multicast receivers. Receivers join a tree rooted at the RP. Senders register their existence with the RP. Initially, traffic from the sender flows through the RP to reach each receiver.

The benefit of PIM-SM is that unlike DVMRP and PIM-DM networks, multicast data is blocked from a network segment unless a downstream device specifically asks to receive the data. This has the potential to significantly reduce the amount of traffic traversing the network. It also implies that no pruning information is maintained for locations with no receivers. This information is maintained only in devices connected to the multicast delivery tree. Because of these benefits, PIM-SM is currently the most popular multicast routing protocol used in the Internet.

Archived

Building the PIM-SM multicast delivery tree

The basic PIM-SM interaction with the RP is:

1. A multicast router sends periodic join messages to a group-specific RP. Each router along the path toward the RP builds and sends join requests to the RP. This builds a group-specific multicast delivery tree rooted at the RP. Like other multicast protocols, the tree is actually a reverse path tree, because join requests follow a reverse path from the receiver to the RP. Figure 6-10 shows this function.

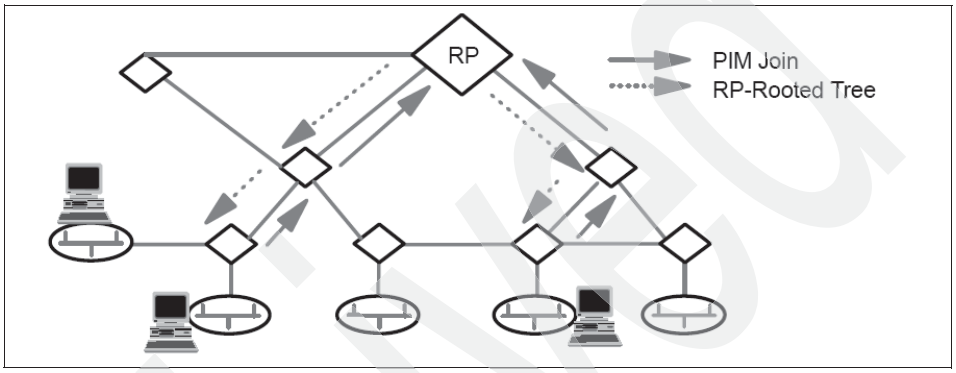


Figure 6-10 Creating the RP-rooted delivery tree

2. The multicast router connecting to the source initially encapsulates each multicast packet in a register message. These messages are sent to the RP. The RP decapsulates these unicast messages and forwards the data packets to the set of downstream receivers. Figure 6-11 shows this function.

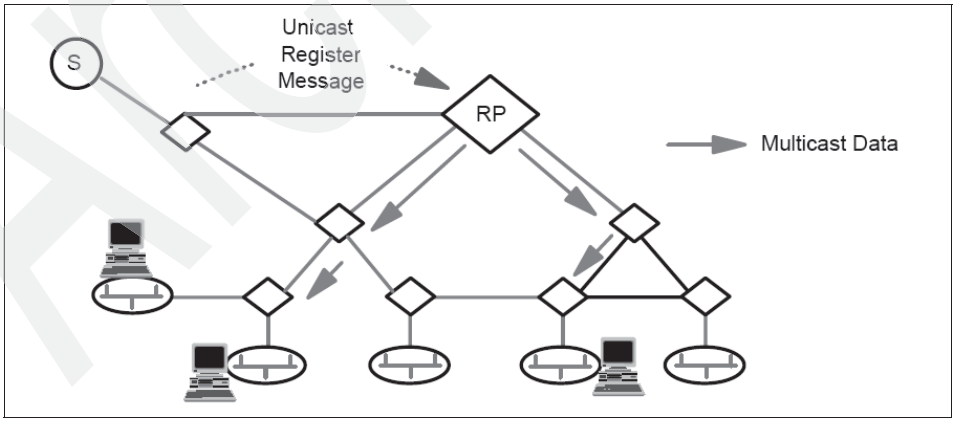


Figure 6-11 Registering a source

3. The RP-based delivery tree can reflect suboptimal routes to some receivers. To optimize these connections, the router can create a *source-based* multicast delivery tree. Figure 6-12 shows this function.

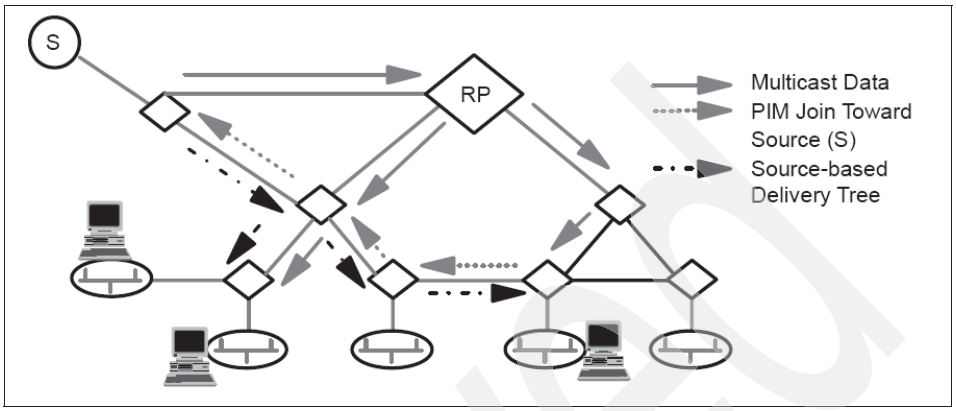


Figure 6-12 Establishing a source-based delivery tree

4. After the router receives multicast packets through both the source-based delivery tree and the RP-based delivery tree, PIM prune messages are sent toward the RP to prune this branch of the tree. When complete, multicast data from the source flows only through the source-based delivery tree. Figure 6-13 shows this function.

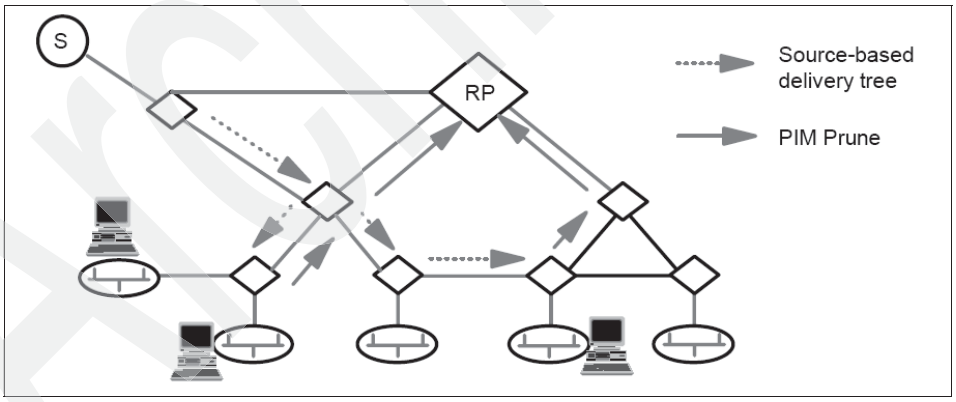


Figure 6-13 Eliminating the RP-based delivery tree

The switch to the source-based tree occurs after a significant number of packets have been received from the source. To implement this policy, the router monitors the quantity of packets received through the RP-based delivery tree. When this data rate exceeds a configured threshold, the device initiates the change.

RP selection

An RP is selected as part of standard PIM-SM operations. An RP is mapped to each specific multicast group. To perform this mapping, a router configured to support PIM-SM distributes a list of candidate RPs to all other routers in the environment. When a mapping needs to be performed, each router hashes the multicast group address into an IP address that represents the RP.

PIM-SM benefits

PIM-SM is optimized for environments containing a large number of multicast data streams. Each stream should flow to a relatively small number of the LAN segments. For these groups, the flooding and pruning associated with PIM-DM and DVMRP is an inefficient use of network bandwidth. PIM-SM is also appropriate when:

- There are few receivers in a group.
- Senders and receivers are separated by WAN links.
- The stream of multicast traffic is intermittent.

Like PIM-DM, PIM-SM assumes that the route obtained through the unicast routing protocol supports multicast routing. Therefore, the network administrator must ensure that each device connected to the end-to-end path is multicast-enabled.

6.8 Interconnecting multicast domains

Early multicast development focused on a flat network topology. This differed from the hierarchical topology deployed in the Internet. Like the Internet, multicast developers soon realized the need to perform inter-domain routing. New classes of protocol have been proposed to address this deficiency. There are currently two approaches to interconnecting multicast domains.

6.8.1 Multicast Source Discovery Protocol (MSDP)

MSDP, defined in RFC 3618, is a protocol to logically connect multiple PIM-SM domains. It is used to find active multicast sources in other domains. RPs in separate autonomous systems communicate through MSDP to exchange information about these multicast sources. Each domain continues to use its own RP. There is no direct dependence on other RPs for intra-domain communication.

Typically, each AS will contain one MSDP-speaking RP. Other autonomous systems create MSDP peer sessions with this RP. These sessions are used to exchange the lists of source speakers in each specific multicast group.

MSDP is not directly involved in multicast data delivery. Its purpose is to discover sources in other domains. If a device in one domain wants to receive data from a source in another domain, the data is delivered using the standard operations within PIM-SM.

MSDP peers will usually be separated by multiple hops. All devices used to support this remote communication must be multicast capable.

MSDP relies heavily on the Multiprotocol Extensions to BGP (MBGP) for interdomain communication. For an MBGP overview, see “Multiprotocol extensions for BGP-4” on page 268.

MSDP operations

To establish multicast communications between a source in one domain and receivers in other domains, MSDP uses the following steps:

1. Following standard PIM-SM operations, the DR for the source sends a PIM register message to the RP. That data packet is decapsulated by the RP and forwarded down the shared tree to receivers in the same domain.
2. The packet is also re-encapsulated in a *source-active (SA)* message. This message is sent to all MSDP peers. The SA message identifies the source and group. This operation occurs when the source becomes active.
3. When an RP for a domain receives an SA message from a MSDP peer, it determines if it has any members interested in the group described by the SA message. If there is an interested party, the RP triggers a join toward the source. After the path has been established and the RP is forwarding data, the receiver can switch to the shortest-path tree directly to the source. This is done with standard PIM-SM conventions. Each MSDP peer receives and forwards SA messages. Each peer examines the MBGP routing table to determine the next-hop peer towards the originating RP. The router forwards the SA message to all MSDP peers other than the RPF peer.
4. If the router receives the SA message from a peer that is not the next-hop peer, the message is discarded. This floods the announcements through the entire network. It is similar to the RPF processing described in 6.4.1, “Reverse path forwarding algorithm” on page 252.

See Figure 6-14 for an overview of MSDP operations.

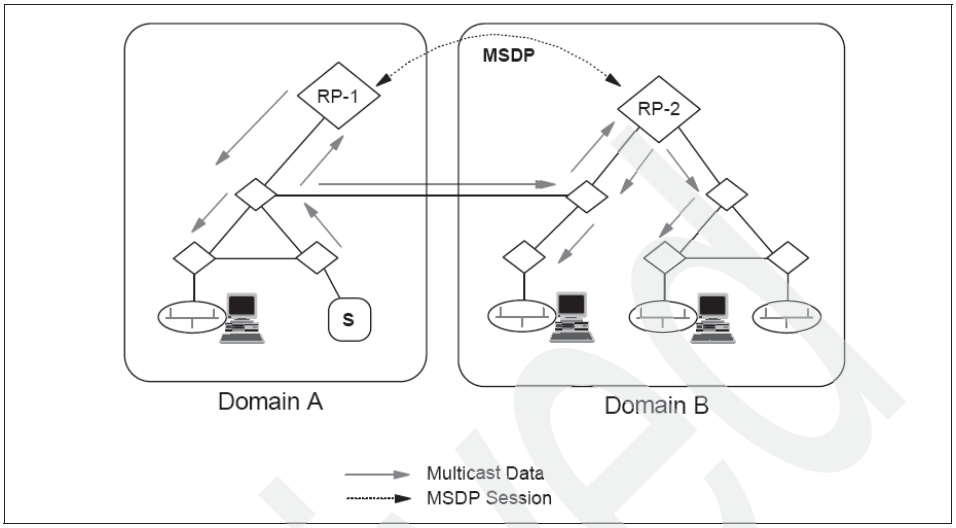


Figure 6-14 MSDP operations

MSDP limitations

MSDP is currently an experimental RFC, but is deployed in numerous network environments. Because of the periodic flood and prune messages associated with MSDP, this protocol does not scale to the address the potential needs of the Internet. It is expected that MSDP will be replaced with the Border Gateway Multicast Protocol (BGMP). We review this protocol 6.8.2, “Border Gateway Multicast Protocol” on page 269.

Multiprotocol extensions for BGP-4

When interconnecting multicast domains, it is possible that unicast routing might select a path containing devices that do not support multicast traffic. This results in multicast join messages not reaching the intended destination.

To solve this problem, MSDP uses the multiprotocol extensions for BGP-4 (MBGP) defined in RFC 2858. This is a set of extensions allowing BGP to maintain separate routing tables for different protocols. Therefore, MBGP can create routes for both unicast and multicast traffic. The multicast routes can traverse around the portions of the environment that do not support multicast. It permits links to be dedicated to multicast traffic. Alternatively, it can limit the resources used to support each type of traffic.

The information associated with the multicast routes is used by PIM to build distribution trees. The standard services for filtering and preference setting are available with MBGP.

6.8.2 Border Gateway Multicast Protocol

The Border Gateway Multicast Protocol (BGMP), defined in RFC 3913, is a multicast routing protocol that builds shared domain trees. Like PIM-SM, BGMP chooses a global root for the delivery tree. However, in BGMP, the root is a domain, not a single router. This allows connectivity to the domain to be maintained whenever any path is available to the domain.

Similar to the cooperation between an Exterior Gateway Protocol (EGP) and an Interior Gateway Protocol (IGP) in a unicast environment, BGMP is used as the inter-domain multicast protocol. Any multicast IGP can be used internally.

BGMP operates between border routers in each domain instead of using an RP. Join messages are used to construct trees between domains. Border routers learn from the multicast IGP whenever a host is interested in participating in an interdomain multicast group. When this occurs, the border router sends a join message to the root domain. Peer devices forward this request towards the root. This forms the shared tree used for multicast delivery.

The BGMP specification requires multicast addresses to be allocated to particular domains. The specification *suggests* the use of the Multicast Address-Set Claim (MASC) protocol, defined in RFC 2909, to achieve this result. MASC is a separate protocol allowing domains to claim temporary responsibility for a range of addresses. The BGMP specification does not mandate the use of MASC.

6.9 The multicast backbone

The Internet multicast backbone (MBONE) was established in March 1992. It was initially deployed to provide hands-on experience with multicast protocols. The first uses provided audio multicasting of IETF meetings. At that time, 20 sites were connected to the backbone. Two years later, simultaneous audio and video transmissions were distributed to more than 500 participants located in 15 countries. Since then, the MBONE has been used to broadcast NASA Space Shuttle missions, rock concerts, and numerous technical conferences. Commercial and private use of the MBONE continues to increase.

The multicast backbone started as a virtual overlay network using much of the physical Internet infrastructure. At that time, multicast routing was not supported in standard routing devices. The first MBONE points-of-presence were UNIX systems configured with the mrouted routing process. Today, the MBONE is still operational, but multicast connectivity is natively included in many Internet routers. Additionally, because multicast ability is a standard feature of IPv6,

MBONE will most likely become obsolete as IPv6 becomes more widely implemented.

6.9.1 MBONE routing

Multicast traffic does not flow to every Internet location. Until that occurs, MBONE will consist of a set of multicast network islands. These islands are interconnected through virtual tunnels. The tunnels bridge through areas that do not support multicast traffic.

A router that needs to send multicast packets to another multicast island encapsulates the packets in unicast packets. These encapsulated packets are transmitted through the standard Internet routers. The destination address contained in the unicast packets is the endpoint of the tunnel. The router at the remote end of the tunnel removes the encapsulation header and forwards the multicast packets to the receiving devices.

Figure 6-15 shows an overview of an MBONE tunnel's structure.

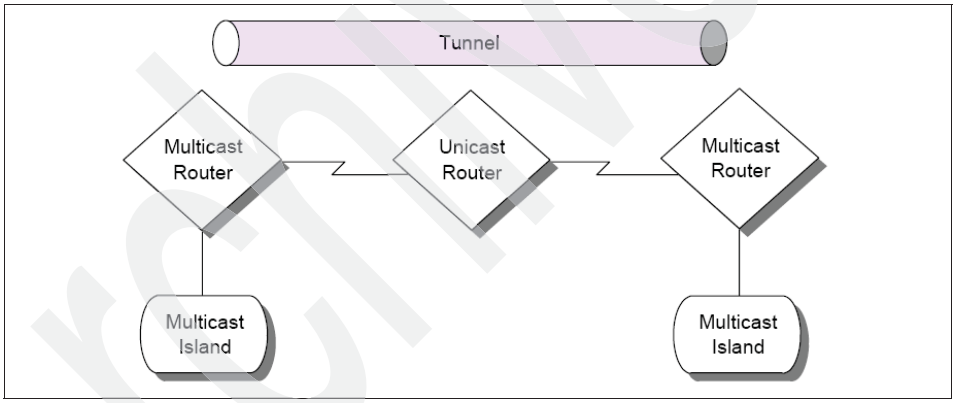


Figure 6-15 MBONE tunnel

MBONE tunnels have associated metric and threshold parameters. The metric parameter is used as a cost in the multicast routing algorithm. The routing algorithm uses this value to select the best path through the network. Figure 6-16 on page 271 depicts an environment containing four multicast sites interconnected through MBONE tunnels. The tunnels have been assigned different metric values to skew traffic forwarding through the network.

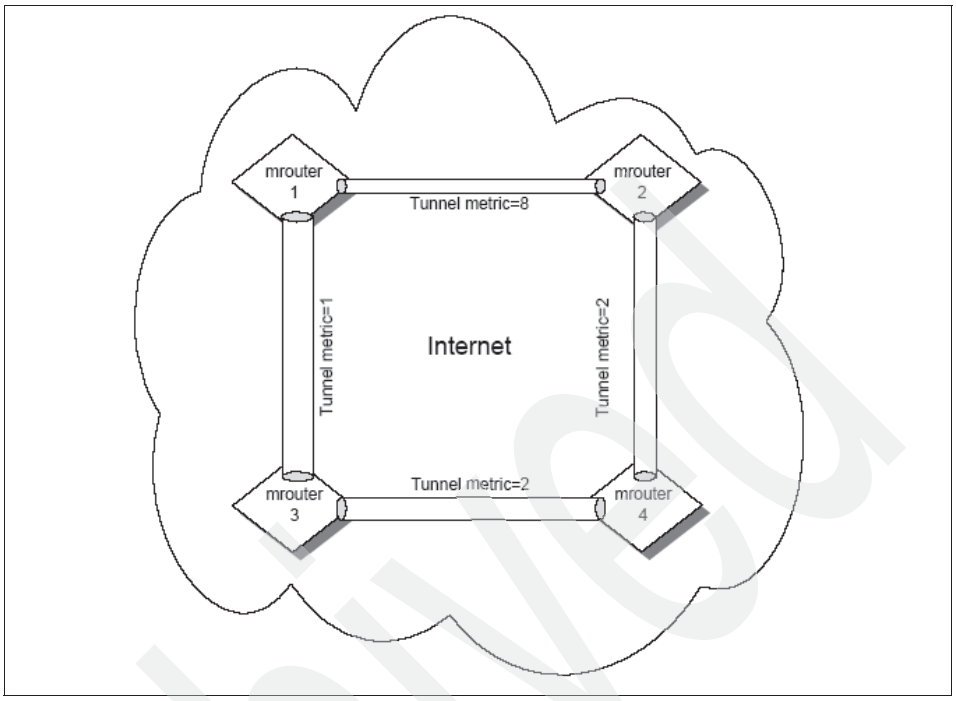


Figure 6-16 MBONE tunnel metric

A multicast packet sent from router 1 to router 2 should not use the tunnel directly connecting router 1 and router 2. The cost of the alternate path using router 3 and router 4 is 5 (1 + 2 + 2). This is more attractive that the direct path between router 1 and router 2; this path has a cost of 8.

The threshold parameter limits the distribution of multicast packets. It specifies a minimum TTL for a multicast packet forwarded into an established tunnel. The TTL is decremented by 1 at each multicast router.

In the future, most Internet routers will provide direct support for IP multicast. This will eliminate the need for multicast tunnels. The current MBONE implementation is only a temporary solution. It will become obsolete when multicasting is fully supported in every Internet router.

6.9.2 Multicast applications

The first multicast applications provided audio conferencing functions. These applications have increased in usability and functionality. Recently, development

of multicast systems has accelerated. New and improved applications are being delivered to support:

- Multimedia conferencing: These tools have been used on the MBONE for several years. They support many-to-many audio-only or audio-video communication. When used in conjunction with whiteboard applications, these conferences enhance collaboration while requiring minimal bandwidth.
- Data distribution: These tools provide the ability to simultaneously deliver data to large numbers of receivers. For example, a central site can efficiently push updated data files to each district office.
- Gaming and simulation: These applications have been readily available. However, the integration of multicast services allow the applications to scale to a large number of users. Multicast groups can represent different sections of the game or simulation. As users move from one section to the next, they exit and join different multicast groups.
- Real-time data multicast: These applications distribute real-time data to large numbers of users. For example, stock ticker information can be provided to sets of workstations. The use of multicast groups can tailor the information received by a specific device.

Many of these applications use UDP instead of the usual TCP transport support. With TCP, reliability and flow control mechanisms have not been optimized for real-time broadcasting of multimedia data. Frequently, the potential to lose a small percentage of packets is preferred to the transmission delays introduced with TCP.

In addition to UDP, most applications use the Real-Time Transport Protocol (refer to 21.3.4, “Real-Time Transport Protocol (RTP)” on page 756). This protocol provides mechanisms to continuously transmit multimedia data streams through the Internet without incurring additional delays.

6.10 RFCs relevant to this chapter

The following RFCs provide detailed information about the multicasting protocols and architectures presented throughout this chapter:

- RFC 1075 – Distance Vector Multicast Routing Protocol (November 1988)
- RFC 1112 – Host extensions for IP multicasting (August 1989)
- RFC 1584 – Multicast Extensions to OSPF (March 1994)
- RFC 2236 – Internet Group Management Protocol, Version 2 (November 1997)

- RFC 2362 – Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (June 1998)
- RFC 2858 – Multiprotocol Extensions for BGP-4 (February 1998)
- RFC 2909 – The Multicast Address-Set Claim (MASC) Protocol (September 2000)
- RFC 3232 – Assigned Numbers: RFC 1700 is Replaced by an On-line Database (January 2002)
- RFC 3376 – Internet Group Management Protocol, Version 3 (October 2002)
- RFC 3618 – Multicast Source Discovery Protocol (MSDP) (October 2003)
- RFC 3913 – Border Gateway Multicast Protocol (BGMP): Protocol Specification (September 2004)
- RFC 3973 – Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (January 2005)
- RFC 4541 – Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discover (MLD) Snooping Switches (May 2006)

Archived

Archived

7

Mobile IP

The increasingly mobile nature of the workforce presents problems for the configuration and operation of mobile network devices. It is possible to allocate multiple sets of configuration parameters to a device, but this obviously means an increased workload for the administrator and the user. Perhaps more importantly, however, is that this type of configuration is wasteful with respect to the number of IP addresses allocated.

In DHCP and DDNS environments, DHCP provides a device with a valid IP address for the point at which it is attached to the network. DDNS provides a method of locating that device by its host name, no matter where that device happens to be attached to a network and what IP address it has been allocated. An alternative approach to the problem of dealing with mobile devices is provided in RFC 3344 – IP Mobility Support. IP Mobility Support, commonly referred to as Mobile IP, is a proposed standard, with a status of elective.

© Copyright IBM Corp. 1989-2006. All rights reserved.

7.1 Mobile IP overview

Mobile IP enables a device to maintain the same IP address (its *home address*) wherever it attaches to the network. (Obviously, a device with an IP address plugged into the wrong subnet will normally be unreachable.) However, the mobile device also has a *care-of* address, which connects to the subnet where it is currently located. The care-of address is managed by a *home agent*, which is a device on the home subnet of the mobile device. Any packet addressed to the IP address of the mobile device is intercepted by the home agent and then forwarded to the care-of address through a tunnel. After it arrives at the end of the tunnel, the datagram is delivered to the mobile device. The mobile node generally uses its home address as the source address of all datagrams that it sends.

Mobile IP can help resolve address shortage problems and reduce administrative workload, because each device that needs to attach to the network at multiple locations only requires a single IP address.

The following terminology is used in a mobile IP network configuration:

Home address The static IP address allocated to a mobile node. It does not change, no matter where the node attaches to the network.

Home network A subnet with a network prefix matching the home address of the mobile node. Datagrams intended for the home address of the mobile node will always be routed to this network.

Tunnel The path followed by an encapsulated datagram.

Visited network A network to which the mobile node is connected (other than the node's home network).

Home agent A router on the home network of the mobile node that maintains current location information for the node and tunnels datagrams for delivery to the node when it is away from home.

Foreign agent A router on a visited network that registers the presence of a mobile node and detunnels and forwards datagrams to the node that have been tunneled by the mobile node's home agent.

Archived

7.1.1 Mobile IP operation

Mobility agents (home agents and foreign agents) advertise their presence in the network by means of *agent advertisement* messages, which are ICMP router advertisement messages with extensions (see Figure 7-3 on page 280). A mobile node can also explicitly request one of these messages with an agent solicitation message. When a mobile node connects to the network and receives one of these messages, it is able to determine whether it is on its home network or a foreign network. If the mobile node detects that it is on its home network, it will operate normally, without the use of mobility services. In addition, if it has just returned to the home network, having previously been working elsewhere, it will deregister itself with the home agent. This is done through the exchange of a registration request and registration reply.

If, however, the mobile node detects, from an agent advertisement, that it has moved to a foreign network, it obtains a care-of address for the foreign network. This address can be obtained from the foreign agent (a foreign agent care-of address, which is the address of the foreign agent itself), or it can be obtained by some other mechanism, such as DHCP (in which case, it is known as a co-located care-of address). The use of co-located care-of addresses has the advantage that the mobile node does not need a foreign agent to be present at every network that it visits, but it does require that a pool of IP addresses be made available for visiting mobile nodes by the DHCP server.

Note that communication between a mobile node and a foreign agent takes place at the link layer level. It cannot use the normal IP routing mechanism, because the mobile node's IP address does not belong to the subnet in which it is currently located.

After the mobile node has received its care-of address, it needs to register itself with its home agent. This can be done through the foreign agent, which forwards the request to the home agent, or directly with the home agent (see Figure 7-4 on page 281).

After the home agent has registered the care-of address for the mobile node in its new position, any datagram intended for the home address of the mobile node is intercepted by the home agent and tunneled to the care-of address. The tunnel endpoint can be at a foreign agent (if the mobile node has a foreign agent care-of address), or at the mobile node itself (if it has a co-located care-of address). Here the original datagram is removed from the tunnel and delivered to the mobile node.

The mobile node will generally respond to the received datagram using standard IP routing mechanisms.

Figure 7-1 shows a mobile IP operation.

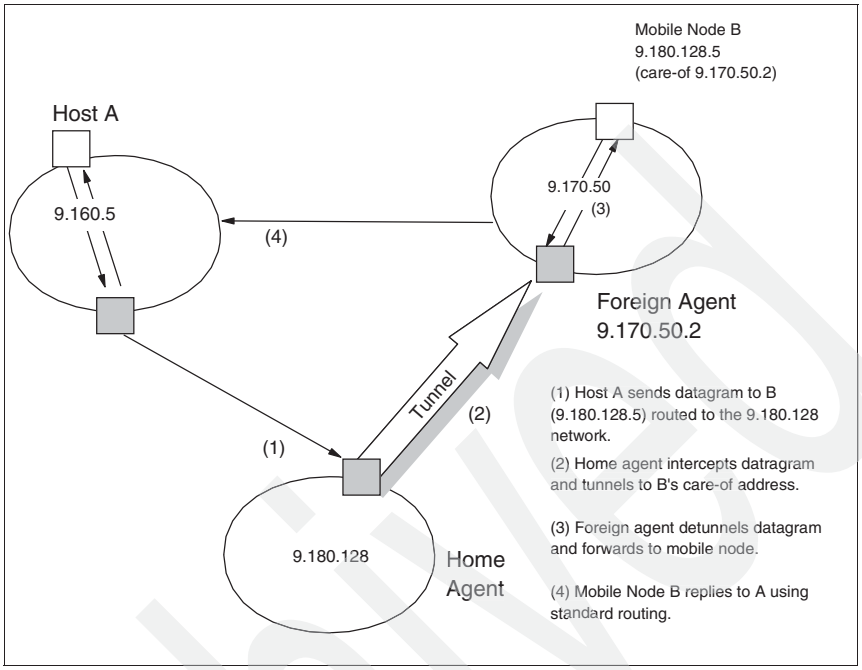


Figure 7-1 Mobile IP operation

7.1.2 Mobility agent advertisement extensions

The mobility agent advertisement consists of an ICMP router Advertisement with one or more of the following extensions, as shown in Figure 7-2.

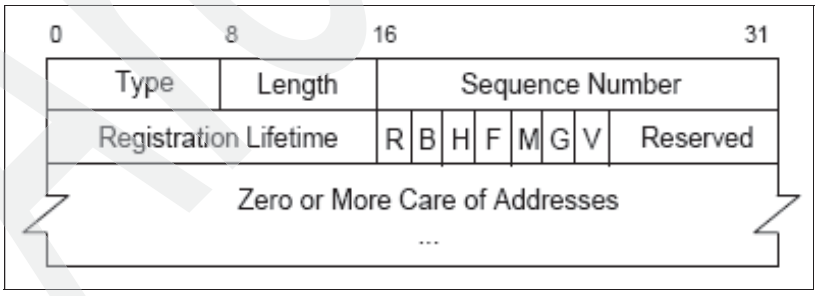


Figure 7-2 Mobility agent advertisement extension

Where:

Type	16.
Length	(6+[4*N]), where N is the number of care-of addresses advertised.
Sequence number	The number of advertisements sent by this agent since it was initialized.
Registration lifetime	The longest lifetime, in seconds, that this agent will accept a Registration Request. A value of 0xffff indicates infinity. This field bears no relationship with the lifetime field in the router advertisement itself.
R	Registration required. Mobile node must register with this agent rather than use a co-located care-of address.
B	Busy. Foreign agent cannot accept additional registrations.
H	Home agent. This agent offers service as a home agent on this link.
F	Foreign agent. This agent offers service as a foreign agent on this link.
M	Minimal encapsulation. This agent receives tunneled datagrams that use minimal encapsulation.
G	GRE encapsulation. This agent receives tunneled datagrams that use GRE encapsulation.
V	Van Jacobson Header Compression. This agent supports use of Van Jacobson Header Compression over the link with any registered mobile node.
Reserved	This area is ignored.
Care-of Address(es)	The care-of address or addresses advertised by this agent. At least one must be included if the F bit is set.

Note that a foreign agent might be too busy to service additional mobile nodes at certain times. However, it must continue to send agent advertisements (with the B bit set) so that mobile nodes that are already registered will know that the agent has not failed or that they are still in range of the foreign agent.

Archived

The prefix lengths extension can follow the mobility agent advertisement extension. It is used to indicate the number of bits that need to be applied to each router address (in the ICMP router advertisement portion of the message) when network prefixes are being used for move detection. See Figure 7-3 for more details.

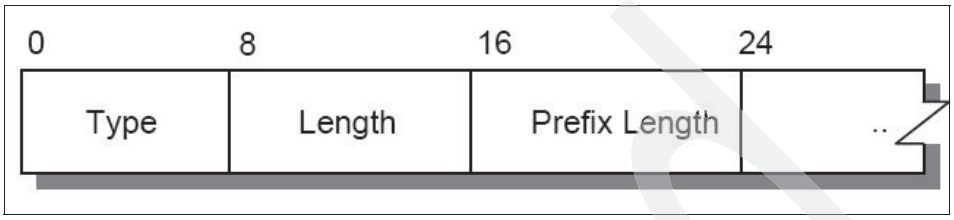


Figure 7-3 Prefix-lengths extensions

Where:

Type 19.

Length The number of router address entries in the router advertisement portion of the agent advertisement.

Prefix length(s) The number of leading bits that make up the network prefix for each of the router addresses in the router advertisement portion of the agent advertisement. Each prefix length is a separate byte, in the order that the router addresses are listed.

7.2 Mobile IP registration process

RFC 3344 defines two different procedures for mobile IP registration: The mobile node can register through a foreign agent, which relays the registration to the mobile node's home agent, or it can register directly with its home agent. The following rules are used to determine which of these registration processes is used:

- If the mobile node has obtained its care-of address from a foreign agent, it must register through that foreign agent.
- If the mobile node is using a co-located care-of address, but has received an agent advertisement from a foreign agent on this subnet (which has the R bit (registration required) set in that advertisement), it registers through the agent. This mechanism allows for accounting to take place on foreign subnets, even if DHCP and co-located care-of address is the preferred method of address allocation.

Archived

- If the mobile node is using a co-located care-of address but has not received such an advertisement, it must register directly with its home agent.
- If the mobile node returns to its home network, it must deregister directly with its home agent.

The registration process involves the exchange of registration request and registration reply messages, which are UDP datagrams. The registration request is sent to port 434. The request consists of a UDP header, followed by the fields shown in Figure 7-4.

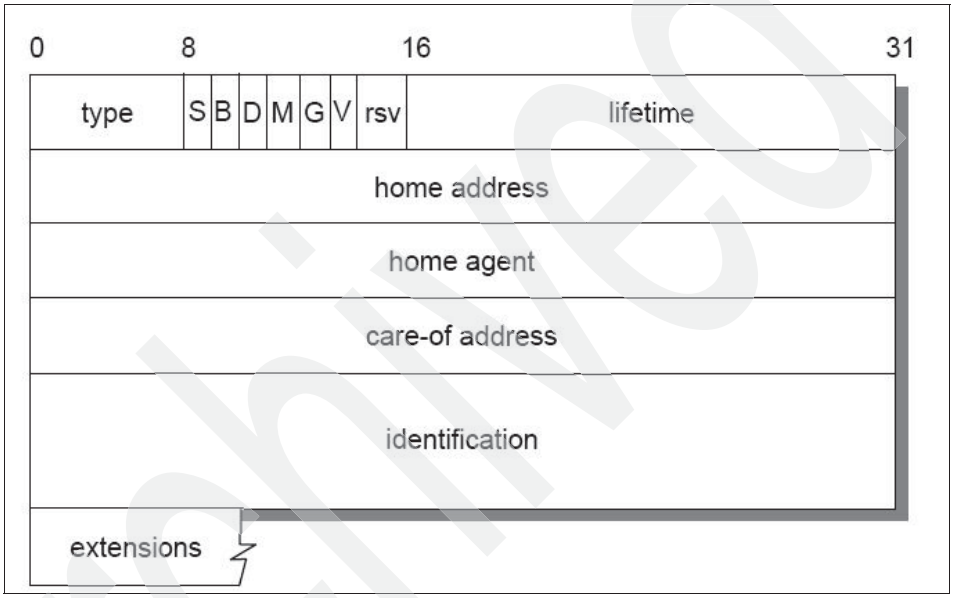


Figure 7-4 Mobile IP: Registration request

Where:

Type 1.

S Simultaneous bindings. If this bit is set, the home agent keeps any previous bindings for this node as well as adding the new binding. The home agent will then forward any datagrams for the node to multiple care-of addresses. This capability is particularly intended for wireless mobile nodes.

B Broadcast datagrams. If this bit is set, the home agent tunnels any broadcast datagrams on the home network to the mobile node.

D	Decapsulation by mobile node. The mobile node is using a co-located care-of address and will, itself, decapsulate the datagrams sent to it.
M	Minimal encapsulation should be used for datagrams tunneled to the mobile node.
G	GRE encapsulation should be used for datagrams tunneled to the mobile node.
V	Van Jacobson compression should be used over the link between agent and mobile node.
rsv	Reserved bits. Sent as zero.
Lifetime	The number of seconds remaining before the registration will be considered expired. A value of zero indicates a request for deregistration. 0xffff indicates infinity.
Home address	The home IP address of the mobile node.
Home agent	The IP address of the mobile node's home agent.
Care-of address	The IP address for the end of the tunnel.
Identification	A 64-bit identification number constructed by the mobile node and used for matching registration requests with replies.
Extensions	A number of extensions are defined, all relating to authentication of the registration process. See RFC 3344 for full details.

Archived

The mobility agent responds to a registration request with a registration reply and with a destination port copied from the source port of the registration request. Figure 7-5 shows the registration reply format.

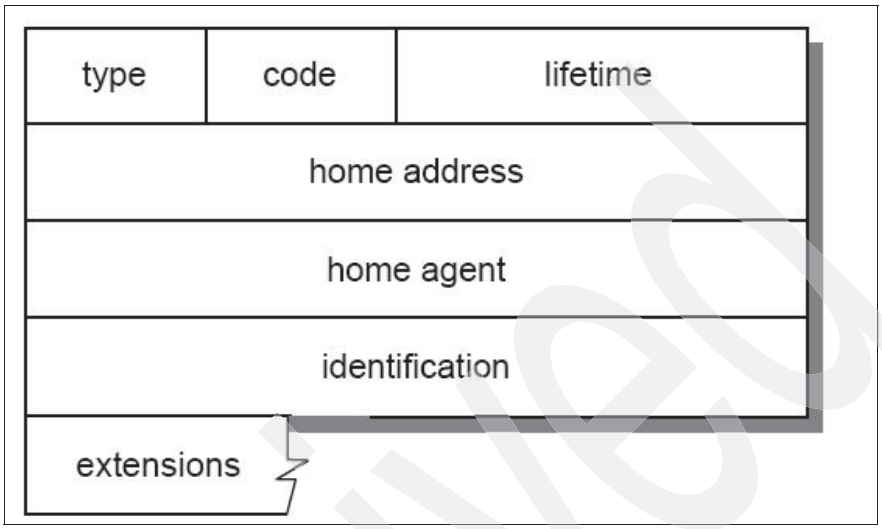


Figure 7-5 Mobile IP: Registration reply

Where:

Type 3.

Code Indicates the result of the registration request:

- **0** Registration accepted.
- **1** Registration accepted, but simultaneous bindings unsupported.
- **64-88** Registration denied by foreign agent.
- **128-136** Registration denied by home agent.

Lifetime The number of seconds remaining before the registration is considered expired. (Code field must be 0 or 1.)

Home address Home IP address of the mobile node.

Home agent IP address of the mobile node's home agent.

Identification A 64-bit identification number used for matching registration requests with replies.

Extensions A number of extensions are defined, all relating to authentication of the registration process.

For full details of these messages, refer to RFC 3344.

Archived

7.2.1 Tunneling

The home agent examines the destination IP address of all datagrams arriving on the home network. If the address matches any of the mobile nodes currently registered as being away from home, the home agent tunnels (using IP in IP encapsulation) the datagram to the care-of address for that mobile node. It is likely that the home agent will also be a router on the home network. In this case, it is likely that it will receive datagrams addressed for a mobile node that is *not* currently registered as being away from home. In this case, the home agent assumes that the mobile node is at home, and forwards the datagram to the home network.

When a foreign agent receives a datagram sent to its advertised care-of address, it compares the inner destination address with its list of registered visitors. If it finds a match, the foreign agent forwards the decapsulated datagram to the appropriate mobile node. If there is no match, the datagram is discarded. (The foreign agent must not forward such a datagram to the original IP header; otherwise, a routing loop occurs.)

If the mobile node is using a co-located care-of address, the end of the tunnel lies at the mobile node itself. The mobile node is responsible for decapsulating the datagrams received from the home agent.

7.2.2 Broadcast datagrams

If the home agent receives a broadcast datagram, it must not forward it to mobile nodes unless the mobile node specifically requested forwarding of broadcasts in its registration request. In this case, it forwards the datagram in one of the following manners:

- If the mobile node has a co-located care-of address, the home agent simply encapsulates the datagram and tunnels it directly to the care-of address.
- If the mobile node has a foreign agent care-of address, the home agent first encapsulates the broadcast in a unicast datagram addressed to the home address of the node. It then encapsulates and tunnels this datagram to the care-of address. In this way, the foreign agent, when it decapsulates the datagram, knows to which of its registered mobile nodes it needs to forward the broadcast.

7.2.3 Move detection

Mobile IP is designed not just for mobile users who regularly move from one site to another and attach their mobile computers to different subnets each time, but also for truly dynamic mobile users (for example, users of a wireless connection

from an aircraft). Two mechanisms are defined that allow the mobile node to detect when it has moved from one subnet to another. When the mobile node detects that it has moved, it must re-register with a care-of address on the new foreign network. The two methods of move detection are as follows:

- Foreign agents are consistently advertising their presence in the network by means of agent advertisements. When the mobile node receives an agent advertisement from its foreign agent, it starts a timer based on the lifetime field in the advertisement. If the mobile node has not received another advertisement from the same foreign agent by the time the lifetime has expired, the mobile node assumes that it has lost contact with that agent. If, in the meantime, it has received an advertisement from *another* foreign agent, it immediately attempts registration with the new agent. If it has not received any further agent advertisements, it uses agent solicitation to try and locate a new foreign agent with which to register.
- The mobile node checks whether any newly received agent advertisements are on the same subnet as its current care-of address. If the network prefix is different, the mobile node assumes that it has moved. On expiration of its current care-of address, the mobile node registers with the foreign agent that sent the new agent advertisement.

7.2.4 Returning home

When the mobile node receives an agent advertisement from its own home agent, it knows that it has returned to its home network. Before deregistering with the home agent, the mobile node must configure its routing table for operation on the home subnet.

7.2.5 ARP considerations

Mobile IP requires two extensions to ARP to cope with the movement of mobile nodes. These are:

Proxy ARP	An ARP reply sent by one node on behalf of another that is either unable or unwilling to answer an ARP request on its own behalf
Gratuitous ARP	An ARP packet sent as a local broadcast packet by one node that causes all receiving nodes to update an entry in their ARP cache

When a mobile node is registered as being on a foreign network, its home agent will use proxy ARP in response to any ARP request seeking the mobile node's MAC address. The home agent responds to the request giving its own MAC address.

When a mobile node moves from its home network and registers itself with a foreign network, the home agent does a gratuitous ARP broadcast to update the ARP caches of all local nodes in the network. The MAC address used is again the MAC address of the home agent.

When a mobile node returns to its home network, having been previously registered at a foreign network, gratuitous ARP is again used to update ARP caches of all local nodes, this time with the real MAC address of the mobile node.

7.2.6 Mobile IP security considerations

The mobile computing environment has many potential vulnerabilities with regard to security, particularly if wireless links are in use, which are particularly exposed to eavesdropping. The tunnel between a home agent and the care-of address of a mobile node can also be susceptible to interception unless a strong authentication mechanism is implemented as part of the registration process. RFC 3344 specifies implementation of keyed MD5 for the authentication protocol and advocates the use of additional mechanisms (such as encryption) for environments where total privacy is required.

7.3 RFCs relevant to this chapter

The following RFC provides detailed information about the connection protocols and architectures presented throughout this chapter:

RFC 3344 – IP Mobility Support (August 2002)

Archived

8

Quality of service

With the increased use of the IP based networks, including the Internet, there has been a large focus on providing necessary network resources to certain applications. That is, it has become better understood that some applications are more “important” than others, thereby demanding preferential treatment throughout an internetwork. Additionally, applications have different demands, such as real-time requirements of low latency and high bandwidth.

This chapter discusses the topic of traffic prioritization, or quality of service (QoS). It explains why QoS can be desirable in an intranet, as well as in the Internet, and presents the two main approaches to implementing QoS in TCP/IP networks:

- Integrated Services
- Differentiated Services

© Copyright IBM Corp. 1989-2006. All rights reserved.

8.1 Why QoS?

In the Internet and intranets of today, bandwidth is an important subject. More and more people are using the Internet for private and business purposes. The amount of data that is being transmitted through the Internet is increasing exponentially. Multimedia applications, such as IP telephony and videoconferencing systems, need a lot more bandwidth than the applications that were used in the early years of the Internet. While traditional Internet applications, such as WWW, FTP, or Telnet, cannot tolerate packet loss but are less sensitive to variable delays, most real-time applications show just the opposite behavior, meaning they can compensate for a reasonable amount of packet loss but are usually very critical toward high variable delays.

This means that without any bandwidth control, the quality of these real-time streams depends on the bandwidth that is currently available. Low or unstable bandwidth causes bad quality in real-time transmissions by leading to, for example, dropouts and hangs. Even the quality of a transmission using the Real-Time Protocol (RTP) depends on the utilization of the underlying IP delivery service.

Therefore, certain concepts are necessary to guarantee a specific quality of service (QoS) for real-time applications on the Internet. A QoS can be described as a set of parameters that describe the quality (for example, bandwidth, buffer usage, priority, and CPU usage) of a specific stream of data. The basic IP protocol stack provides only one QoS, which is called *best-effort*. Packets are transmitted from point to point without any guarantee for a special bandwidth or minimum time delay. With the best-effort traffic model, Internet requests are handled with the *first come, first serve* strategy. This means that all requests have the same priority and are handled one after the other. There is no possibility to make bandwidth reservations for specific connections or to raise the priority for special requests. Therefore, new strategies were developed to provide predictable services for the Internet.

Today, there are two main rudiments to bring QoS to the Internet and IP-based internetworks: Integrated Services and Differentiated Services.

Integrated Services

Integrated Services bring enhancements to the IP network model to support real-time transmissions and guaranteed bandwidth for specific flows. In this case, we define a *flow* as a distinguishable stream of related datagrams from a unique sender to a unique receiver that results from a single user activity and requires the same QoS.

For example, a flow might consist of one video stream between a given host pair. To establish the video connection in both directions, two flows are necessary. Each application that initiates data flows can specify which QoSs are required for this flow. If the videoconferencing tool needs a minimum bandwidth of 128 kbps and a minimum packet delay of 100 ms to assure a continuous video display, such a QoS can be reserved for this connection.

Differentiated Services

Differentiated Services mechanisms do not use per-flow signaling, and as a result, do not consume per-flow state within the routing infrastructure. Different service levels can be allocated to different groups of users, which means that all traffic is distributed into groups or classes with different QoS parameters. This reduces the amount of maintenance required in comparison to Integrated Services.

8.2 Integrated Services

The Integrated Services (IS) model was defined by an IETF working group to be the keystone of the planned IS Internet. This Internet architecture model includes the currently used best-effort service and the new real-time service that provides functions to reserve bandwidth on the Internet and internetworks. IS was developed to optimize network and resource utilization for new applications, such as real-time multimedia, which requires QoS guarantees. Because of routing delays and congestion losses, real-time applications do not work very well on the current best-effort Internet. Video conferencing, video broadcast, and audio conferencing software need guaranteed bandwidth to provide video and audio of acceptable quality. Integrated Services makes it possible to divide the Internet traffic into the standard best-effort traffic for traditional uses and application data flows with guaranteed QoS.

To support the Integrated Services model, an Internet router must be able to provide an appropriate QoS for each flow, in accordance with the service model. The router function that provides different qualities of service is called *traffic control*. It consists of the following components:

Packet scheduler The packet scheduler manages the forwarding of different packet streams in hosts and routers, based on their service class, using queue management and various scheduling algorithms. The packet scheduler must ensure that the packet delivery corresponds to the QoS parameter for each flow. A scheduler can also police or shape the traffic to conform to a certain level of service. The packet scheduler must be implemented at the point

where packets are queued. This is typically the output driver level of an operating system and corresponds to the link layer protocol.

Packet classifier

The packet classifier identifies packets of an IP flow in hosts and routers that will receive a certain level of service. To realize effective traffic control, each incoming packet is mapped by the classifier into a specific class. All packets that are classified in the same class get the same treatment from the packet scheduler. The choice of a class is based on the source and destination IP address and port number in the existing packet header or an additional classification number, which must be added to each packet. A class can correspond to a broad category of flows.

For example, all video flows from a video conference with several participants can belong to one service class. But it is also possible that only one flow belongs to a specific service class.

Admission control

The admission control contains the decision algorithm that a router uses to determine if there are enough routing resources to accept the requested QoS for a new flow. If there are not enough free routing resources, accepting a new flow would impact earlier guarantees and the new flow must be rejected. If the new flow is accepted, the reservation instance in the router assigns the packet classifier and the packet scheduler to reserve the requested QoS for this flow. Admission control is invoked at each router along a reservation path to make a local accept/reject decision at the time a host requests a real-time service. The admission control algorithm must be consistent with the service model.

Admission control is sometimes confused with policy control, which is a packet-by-packet function, processed by the packet scheduler. It ensures that a host does not violate its promised traffic characteristics. Nevertheless, to ensure that QoS guarantees are honored, the admission control will be concerned with enforcing administrative policies on resource reservations. Some policies will be used to check the user authentication for a requested reservation. Unauthorized reservation requests can be rejected. As a result, admission control can play an important role in accounting costs for Internet resources in the future.

Figure 8-1 shows the operation of the Integrated Service model within a host and a router.

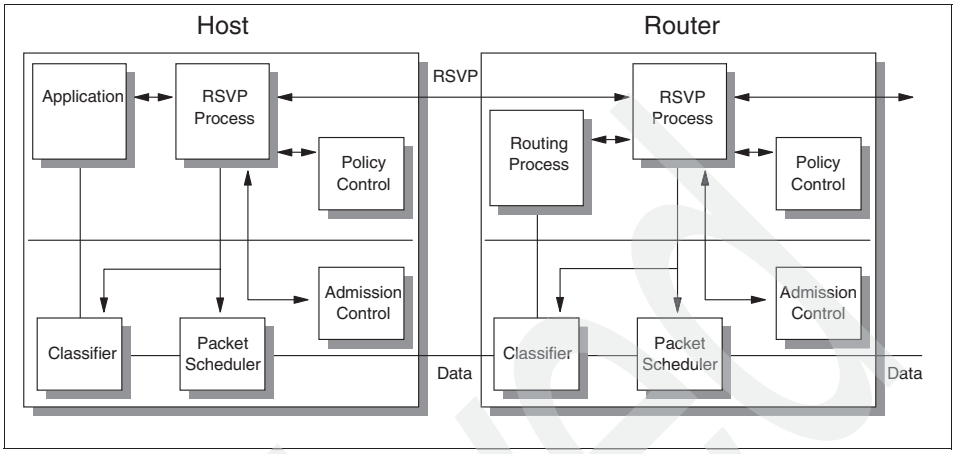


Figure 8-1 The Integrated Service model

Integrated Services use the Resource Reservation Protocol (RSVP) for the signalling of the reservation messages. The IS instances communicate through RSVP to create and maintain flow-specific states in the endpoint hosts and in routers along the path of a flow. See 8.2.4, “The Resource Reservation Protocol (RSVP)” on page 296 for a detailed description of the RSVP protocol. As shown in Figure 8-1, the application that wants to send data packets in a reserved flow communicates with the reservation instance RSVP. The RSVP protocol tries to set up a flow reservation with the requested QoS, which will be accepted if the application fulfilled the policy restrictions and the routers can handle the requested QoS. RSVP advises the packet classifier and packet scheduler in each node to process the packets for this flow adequately. If the application now delivers the data packets to the classifier in the first node, which has mapped this flow into a specific service class complying the requested QoS, the flow is recognized with the sender IP address and is transmitted to the packet scheduler. The packet scheduler forwards the packets, dependent on their service class, to the next router or, finally, to the receiving host. Because RSVP is a simplex protocol, QoS reservations are only made in one direction, from the sending node to the receiving node. If the application in our example wants to cancel the reservation for the data flow, it sends a message to the reservation instance, which frees the reserved QoS resources in all routers along the path. The resources can then be used for other flows. The IS specifications are defined in RFC 1633.

8.2.1 Service classes

The Integrated Services model uses different classes of service that are defined by the Integrated Services IETF working group. Depending on the application, those service classes provide tighter or looser bounds on QoS controls. The current IS model includes the *Guaranteed Service*, which is defined in RFC 2212, and the *Controlled Load Service*, which is defined in RFC 2211. To understand these service classes, some terms need to be explained. Because the IS model provides per-flow reservations, each flow is assigned a flow descriptor. The flow descriptor defines the traffic and QoS characteristics for a specific flow of data packets. In the IS specifications, the flow descriptor consists of a filter specification (filterspec) and a flow specification (flowspec), as illustrated in Figure 8-2.

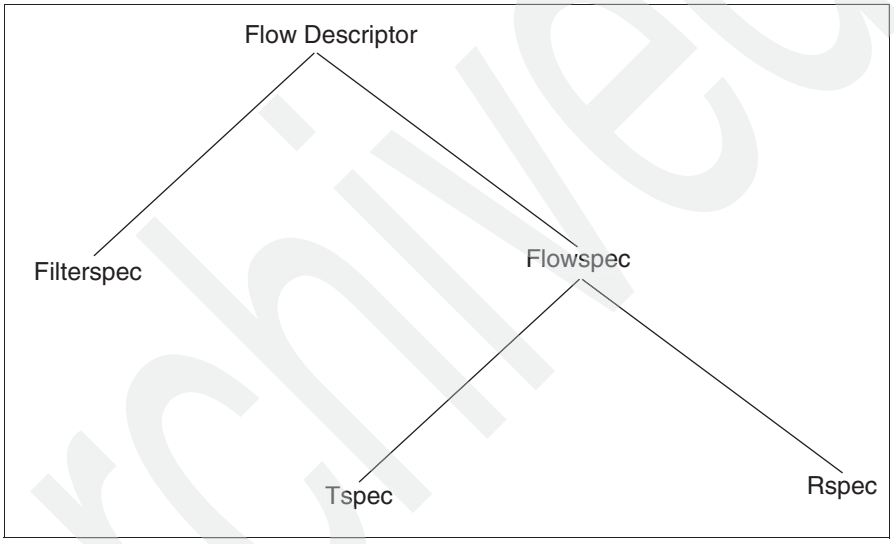


Figure 8-2 Flow descriptor

The filterspec identifies the packets that belong to a specific flow with the sender IP address and source port. The information from the filterspec is used in the packet classifier. The flowspec contains a set of parameters that are called the *invocation information*. The invocation information divides into two groups:

- Traffic Specification (Tspec)
- Service Request Specification (Rspec)

The Tspec describes the traffic characteristics of the requested service. In the IS model, this Tspec is represented with a *token bucket filter*. This principle defines a data-flow control mechanism that adds characters (tokens) in periodical time intervals into a buffer (bucket) and allows a data packet to leave the sender only if

there are at least as many tokens in the bucket as the packet length of the data packet. This strategy allows precise control of the time interval between two data packets in the network. The token bucket system is specified by two parameters: the *token rate r*, which represents the rate at which tokens are placed into the bucket, and the *bucket capacity b*. Both r and b must be positive. Figure 8-3 illustrates the token bucket model.

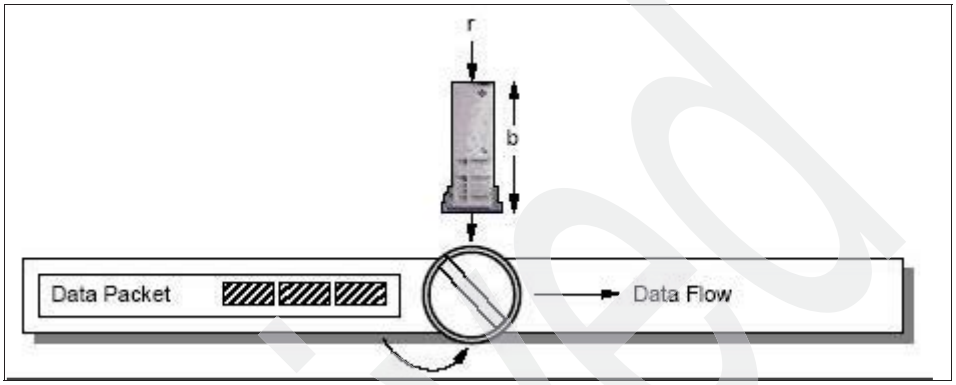


Figure 8-3 Token bucket filter

The parameter *r* specifies the long-term data rate and is measured in bytes of IP datagrams per second. The value of this parameter can range from 1 byte per second to 40 terabytes per second. The parameter *b* specifies the burst data rate allowed by the system and is measured in bytes. The value of this parameter can range from 1 byte to 250 gigabytes. The range of values allowed for these parameters is intentionally large in order to be prepared for future network technologies. The network elements are not expected to support the full range of the values. Traffic that passes the token bucket filter must obey the rule that over all time periods T (seconds), the amount of data sent does not exceed rT+b, where r and b are the token bucket parameters.

Two other token bucket parameters are also part of the Tspec. The *minimum policed unit* m and the *maximum packet size* M. The parameter m specifies the minimum IP datagram size in bytes. Smaller packets are counted against the token bucket filter as being of size m. The parameter M specifies the maximum packet size in bytes that conforms to the Tspec. Network elements must reject a service request if the requested maximum packet size is larger than the MTU size of the link. Summarizing, the token bucket filter is a policing function that isolates the packets that conform to the traffic specifications from the ones that do not conform.

The Service Request Specification (Rspec) specifies the quality of service the application wants to request for a specific flow. This information depends on the type of service and the needs of the QoS requesting application. It can consist of

a specific bandwidth, a maximum packet delay, or a maximum packet loss rate. In the IS implementation, the information from Tspec and Rspec is used in the packet scheduler.

8.2.2 Controlled Load Service

The Controlled Load Service is intended to support the class of applications that are highly sensitive to overloaded conditions in the Internet, such as real-time applications. These applications work well on underloaded networks, but degrade quickly under overloaded conditions. If an application uses the Controlled Load Service, the performance of a specific data flow does not degrade if the network load increases.

The Controlled Load Service offers only one service level, which is intentionally minimal. There are no optional features or capabilities in the specification. The service offers only a single function. It approximates best-effort service over lightly loaded networks. This means that applications that make QoS reservations using Controlled Load Services are provided with service closely equivalent to the service provided to uncontrolled (best-effort) traffic under lightly loaded conditions. In this context, *lightly loaded conditions* means that a very high percentage of transmitted packets will be successfully delivered to the destination, and the transit delay for a very high percentage of the delivered packets will not greatly exceed the minimum transit delay.

Each router in a network that accepts requests for Controlled Load Services must ensure that adequate bandwidth and packet processing resources are available to handle QoS reservation requests. This can be realized with active admission control. Before a router accepts a new QoS reservation, represented by the Tspec, it must consider all important resources, such as link bandwidth, router or switch port buffer space, and computational capacity of the packet forwarding.

The Controlled Load Service class does not accept or make use of specific target values for control parameters, such as bandwidth, delay, or loss. Applications that use Controlled Load Services must guard against small amounts of packet loss and packet delays.

QoS reservations using Controlled Load Services need to provide a Tspec that consists of the token bucket parameters r and b, as well as the minimum policed unit m and the maximum packet size M. An Rspec is not necessary, because Controlled Load Services does not provide functions to reserve a fixed bandwidth or guarantee minimum packet delays. Controlled Load Service provides QoS control only for traffic that conforms to the Tspec that was provided at setup time. This means that the service guarantees only apply for packets that

respect the token bucket rule that over all time periods T, the amount of data sent cannot exceed rT+b.

Controlled Load Service is designed for applications that can tolerate a reasonable amount of packet loss and delay, such as audio and videoconferencing software.

8.2.3 Guaranteed Service

The Guaranteed Service model provides functions that assure that datagrams will arrive within a guaranteed delivery time. This means that every packet of a flow that conforms to the traffic specifications will arrive at least at the maximum delay time that is specified in the flow descriptor. Guaranteed Service is used for applications that need a guarantee that a datagram will arrive at the receiver not later than a certain time after it was transmitted by its source.

For example, real-time multimedia applications, such as video and audio broadcasting systems that use streaming technologies, cannot use datagrams that arrive after their proper play-back time. Applications that have hard real-time requirements, such as real-time distribution of financial data (share prices), will also require guaranteed service. Guaranteed Service does not minimize jitter (the difference between the minimal and maximal datagram delays), but it controls the maximum queuing delay.

The Guaranteed Service model represents the extreme end of delay control for networks. Other service models providing delay control have much weaker delay restrictions. Therefore, Guaranteed Service is only useful if it is provided by every router along the reservation path.

Guaranteed Service gives applications considerable control over their delay. It is important to understand that the delay in an IP network has two parts: a fixed transmission delay and a variable queuing delay. The fixed delay depends on the chosen path, which is determined not by guaranteed service but by the setup mechanism. All data packets in an IP network have a minimum delay that is limited by the speed of light and the turnaround time of the data packets in all routers on the routing path. The queuing delay is determined by Guaranteed Service and it is controlled by two parameters: the token bucket (in particular, the bucket size b) and the bandwidth R that is requested for the reservation. These parameters are used to construct the *fluid model* for the end-to-end behavior of a flow that uses Guaranteed Service.

The fluid model specifies the service that would be provided by a dedicated link between sender and receiver that provides the bandwidth R. In the fluid model, the flow's service is completely independent from the service for other flows. The definition of Guaranteed Service relies on the result that the fluid delay of a flow

obeying a token bucket (r,b) and being served by a line with bandwidth R is bounded by b/R as long as R is not less than r. Guaranteed Service approximates this behavior with the service rate R, where R is now a share of bandwidth through the routing path and not the bandwidth of a dedicated line.

In the Guaranteed Service model, Tspec and Rspec are used to set up a flow reservation. The Tspec is represented by the token bucket parameters. The Rspec contains the parameter R, which specifies the bandwidth for the flow reservation. The Guaranteed Service model is defined in RFC 2212.

8.2.4 The Resource Reservation Protocol (RSVP)

The Integrated Services model uses the Resource Reservation Protocol (RSVP) to set up and control QoS reservations. RSVP is defined in RFC 2205 and has the status of a proposed standard. Because RSVP is an Internet control protocol and not a routing protocol, it requires an existing routing protocol to operate. The RSVP protocol runs on top of IP and UDP and must be implemented in all routers on the reservation path. The key concepts of RSVP are flows and reservations.

An RSVP reservation applies for a specific flow of data packets on a specific path through the routers. As described in 8.1, “Why QoS?” on page 288, a flow is defined as a distinguishable stream of related datagrams from a unique sender to a unique receiver. If the receiver is a multicast address, a flow can reach multiple receivers. RSVP provides the same service for unicast and multicast flows. Each flow is identified from RSVP by its destination IP address and destination port. All flows have dedicated a flow descriptor, which contains the QoS that a specific flow requires. The RSVP protocol does not understand the contents of the flow descriptor. It is carried as an opaque object by RSVP and is delivered to the router's traffic control functions (packet classifier and scheduler) for processing.

Because RSVP is a simplex protocol, reservations are only done in one direction. For duplex connections, such as video and audio conferences, where each sender is also a receiver, it is necessary to set up two RSVP sessions for each station.

The RSVP protocol is receiver-initiated. Using RSVP signalling messages, the sender provides a specific QoS to the receiver, which sends an RSVP reservation message back, with the QoS that should be reserved for the flow, from the sender to the receiver. This behavior considers the different QoS requirements for heterogeneous receivers in large multicast groups. The sender does not need to know the characteristics of all possible receivers to structure the reservations.

To establish a reservation with RSVP, the receivers send reservation requests to the senders, depending on their system capabilities. For example, a fast workstation and a slow PC want to receive a high-quality MPEG video stream with 30 frames per second, which has a data rate of 1.5 Mbps. The workstation has enough CPU performance to decode the video stream, but the PC can only decode 10 frames per second. If the video server sends the messages to the two receivers that it can provide the 1.5 Mbps video stream, the workstation can return a reservation request for the full 1.5 Mbps. But the PC does not need the full bandwidth for its flow because it cannot decode all frames. So the PC may send a reservation request for a flow with 10 frames per second and 500 kbps.

RSVP operation

A basic part of a resource reservation is the path. The path is the way of a packet flow through the different routers from the sender to the receiver. All packets that belong to a specific flow will use the same path. The path gets determined if a sender generates messages that travel in the same direction as the flow. Each sender host periodically sends a path message for each data flow it originates. The path message contains traffic information that describes the QoS for a specific flow. Because RSVP does not handle routing by itself, it uses the information from the routing tables in each router to forward the RSVP messages.

Archived

If the path message reaches the first RSVP router, the router stores the IP address from the *last hop* field in the message, which is the address of the sender. Then the router inserts its own IP address into the last hop field, sends the path message to the next router, and the process repeats until the message has reached the receiver. At the end of this process, each router will know the address from the previous router and the path can be accessed backwards. Figure 8-4 shows the process of the path definition.

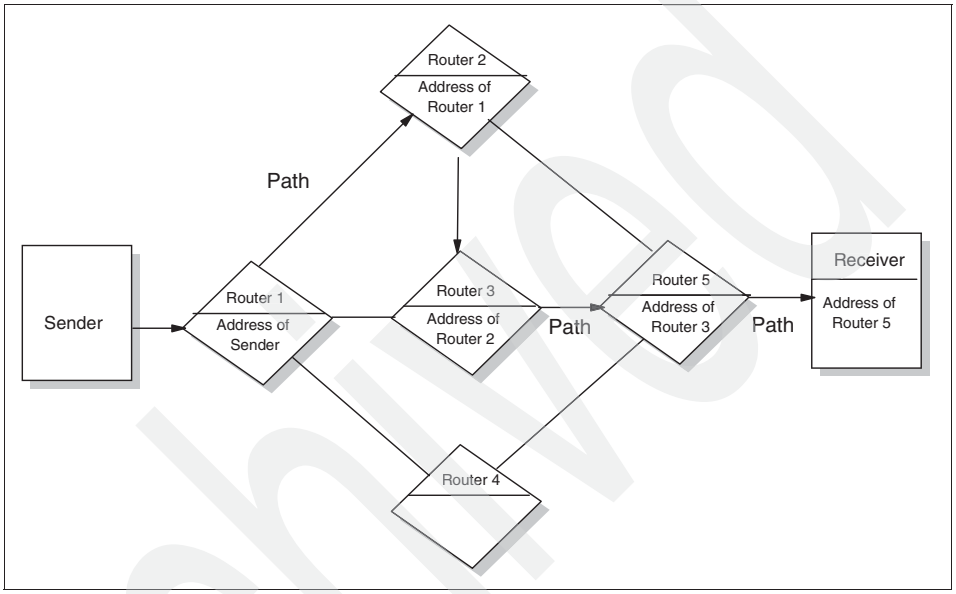


Figure 8-4 RSVP path definition process

Routers that have received a path message are prepared to process resource reservations for a flow. All packets that belongs to this flow will take the same way through the routers (the way that was defined with the path messages).

The status in a system after sending the path messages is as follows: All receivers know that a sender can provide a special QoS for a flow and all routers know about the possible resource reservation for this flow.

Now if a receiver wants to reserve QoS for this flow, it sends a reservation (Resv) message. The reservation message contains the QoS requested from this receiver for a specific flow and is represented by the filterspec and flowspec that form the flow descriptor. The receiver sends the Resv message to the last router in the path with the address it received from the path message. Because every RSVP-capable device knows the address of the previous device on the path, reservation messages travel the path in reverse direction toward the sender and

establish the resource reservation in every router. Figure 8-5 shows the flow of the reservation messages trough the routers.

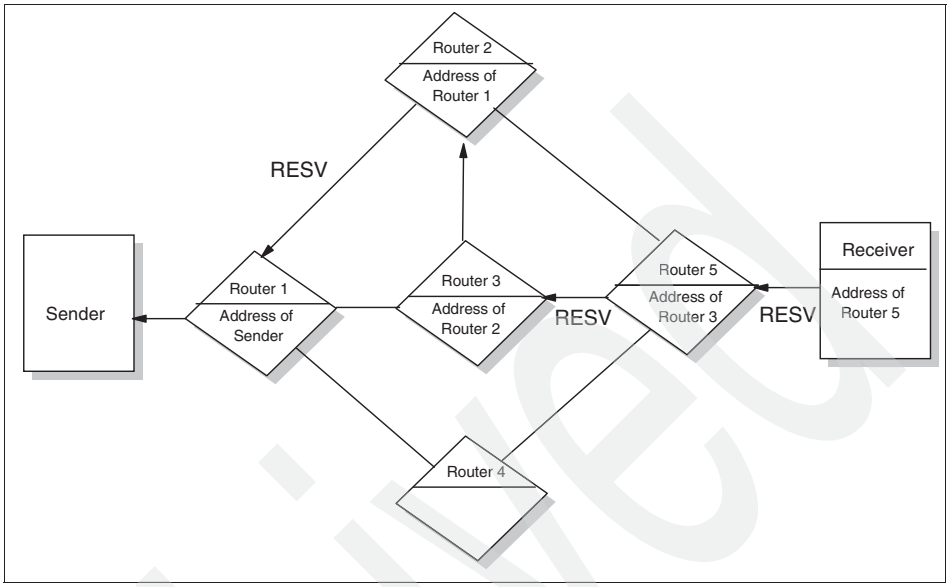


Figure 8-5 RSVP Resv messages flow

At each node, a reservation request initiates two actions:

1. QoS reservation on this link

   The RSVP process passes the request to the admission control and policy control instance on the node. The admission control checks if the router has the necessary resources to establish the new QoS reservation, and the policy control checks if the application has the authorization to make QoS requests. If one of these tests fails, the reservation is rejected and the RSVP process returns a *ResvErr* error message to the appropriate receiver. If both checks succeed, the node uses the filterspec information in the Resv message to set the packet classifier and the flowspec information to set the packet scheduler. After this, the packet classifier will recognize the packets that belong to this flow, and the packet scheduler will obtain the desired QoS defined by the flowspec.

Archived

Figure 8-6 shows the reservation process in an RSVP router.

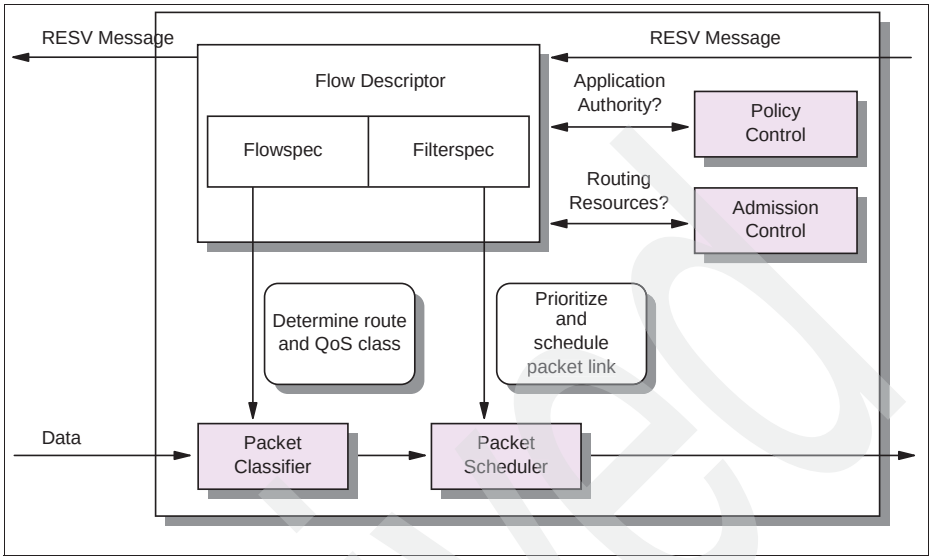


Figure 8-6 RSVP reservation process

2. Forwarding of the reservation request

   After a successful admission and policy check, a reservation request is propagated upstream toward the sender. In a multicast environment, a receiver can get data from multiple senders. The set of sender hosts to which a given reservation request is propagated is called the *scope* of that request. The reservation request that is forwarded by a node after a successful reservation can differ from the request that was received from the previous hop downstream. One possible reason for this is that the traffic control mechanism may modify the flowspec hop-by-hop. Another more important reason is that in a multicast environment, reservations from different downstream branches, but for the same sender, are *merged* together as they travel across the upstream path. This merging is necessary to conserve resources in the routers.

   A successful reservation request propagates upstream along the multicast tree until it reaches a point where an existing reservation is equal or greater than that being requested. At this point, the arriving request is merged with the reservation in place and does not need to be forwarded further.

Figure 8-7 shows the reservation merging for a multicast flow.

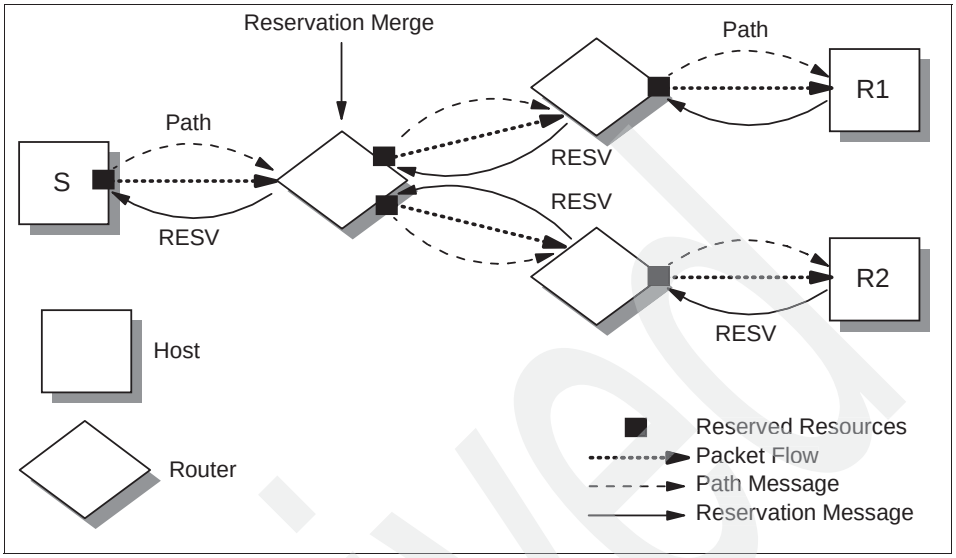


Figure 8-7 RSVP reservation merging for multicast flow

If the reservation request reaches the sender, the QoS reservation was established in every router on the path and the application can start to send packets downstream to the receivers. The packet classifier and the packet scheduler in each router make sure that the packets are forwarded according to the requested QoS.

This type of reservation is only reasonable if all routers on the path support RSVP. If only one router does not support resource reservation, the service cannot be guaranteed for the whole path because of the “best effort” characteristics of normal routers. A router on the path that does not support RSVP is a bottleneck for the flow.

A receiver that originates a reservation request can also request a confirmation message that indicates that the request was installed in the network. The receiver includes a confirmation request in the Resv message and gets a ResvConf message if the reservation was established successfully.

RSVP resource reservations maintain soft-state in routers and hosts, which means that a reservation is canceled if RSVP does not send refresh messages along the path for an existing reservation. This allows route changes without resulting in protocol inefficiencies. Path messages must also be present because the path state fields in the routers will be reset after a timeout period.

Path and reservation states can also be deleted with RSVP *teardown* messages. There are two types of teardown messages:

– PathTear messages

  PathTear messages travel downstream from the point of initiation to all receivers, deleting the path state as well as all dependent reservation states in each RSVP-capable device.

– ResvTear messages

  ResvTear messages travel upstream from the point of initiation to all senders, deleting reservation states in all routers and hosts.

A teardown request can be initiated by senders, receivers, or routers that notice a state timeout. Because of the soft-state principle of RSVP reservations, it is not really necessary to explicitly tear down an old reservation. Nevertheless, we recommend that all end hosts send a teardown request if a consisting reservation is no longer needed.

RSVP reservation styles

Users of multicast multimedia applications often receive flows from different senders. In the reservation process described in “RSVP operation” on page 297, a receiver must initiate a separate reservation request for each flow it wants to receive. But RSVP provides a more flexible way to reserve QoS for flows from different senders. A reservation request includes a set of options that are called the *reservation style*. One of these options deals with the treatment of reservations for different senders within the same session. The receiver can establish a *distinct* reservation for each sender or make a single *shared* reservation for all packets from the senders in one session.

Another option defines how the senders for a reservation request are selected. It is possible to specify an *explicit* list or a *wildcard* that selects the senders belonging to one session. In an explicit sender-selection reservation, a filterspec must identify exactly one sender. In a wildcard sender-selection, the filterspec is not needed. Figure 8-8 shows the reservation styles that are defined with this reservation option.

Sender Selection	Distinct Reservation	Shared Reservation
Explicit	Fixed-Filter (FF) Style	Shared-Explicit (SE) Style
Wildcard	(Not Defined)	Wildcard-Filter (WF) Style

Figure 8-8 RSVP reservation styles

Where:

Wildcard-Filter (WF) The Wildcard-Filter style uses the options shared reservation and wildcard sender selection. This reservation style establishes a single reservation for all senders in a session. Reservations from different senders are merged together along the path so that only the biggest reservation request reaches the senders.

A wildcard reservation is forwarded upstream to all sender hosts. If new senders appear in the session, for example, new members enter a videoconferencing, the reservation is extended to these new senders.

Fixed-Filter (FF) The Fixed-Filter style uses the option's distinct reservations and explicit sender selection. This means that a distinct reservation is created for data packets from a particular sender. Packets from different senders that are in the same session do not share reservations.

Shared-Explicit (SE) The Shared-Explicit style uses the option's shared reservation and explicit sender selection. This means that a single reservation covers flows from a specified subset of senders. Therefore, a sender list must be included into the reservation request from the receiver.

Reservations established in shared style (WF and SE) are mostly used for multicast applications. For this type of application, it is unlikely that several data sources transmit data simultaneously, so it is not necessary to reserve QoS for each sender.

For example, in an audio conference that consists of five participants, every station sends a data stream with 64 kbps. With a Fixed-Filter style reservation, all members of the conference must establish four separate 64 kbps reservations for the flows from the other senders. But in an audio conference, usually only one or two people speak at the same time. Therefore, it is sufficient to reserve a bandwidth of 128 kbps for all senders, because most audio conferencing software uses silence suppression, which means that if a person does not speak, no packets are sent. This can be realized if every receiver makes one shared reservation of 128 kbps for all senders.

Using the Shared-Explicit style, all receivers must explicitly identify all other senders in the conference. With the Wildcard-Filter style, the reservation counts for every sender that matches the reservation specifications. If, for example, the audio conferencing tool sends the data packets to a special TCP/IP port, the receivers can make a Wildcard-Filter reservation for all packets with this destination port.

RSVP message format

An RSVP message basically consists of a common header, followed by a body consisting of a variable number of *objects*. The number and the content of these objects depend on the message type. The message objects contain the information that is necessary to realize resource reservations, for example, the flow descriptor or the reservation style. In most cases, the order of the objects in an RSVP message makes no logical difference. RFC 2205 recommends that an RSVP implementation should use the object order defined in the RFC, but accepts the objects in any permissible order. Figure 8-9 shows the common header of a RSVP message.

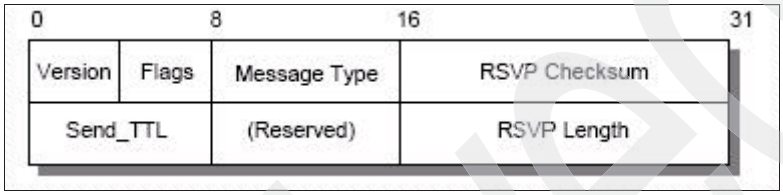


Figure 8-9 RSVP common header

Where:

Version 4-bit RSVP protocol number. The current version is 1.

Flags 4-bit field that is reserved for flags. No flags are defined yet.

Message type 8-bit field that specifies the message type:

- Path
- Resv
- PathErr
- ResvErr
- PathTear
- ResvTear
- ResvConf

RSVP vhecksum 16-bit field. The checksum can be used by receivers of an RSVP message to detect errors in the transmission of this message.

Send_TTL 8-bit field, which contains the IP TTL value with which the message was sent.

RSVP length 16-bit field that contains the total length of the RSVP message including the common header and all objects that follow. The length is counted in bytes.

The RSVP objects that follow the common header consist of a 32-bit header and one or more 32-bit words. Figure 8-10 shows the RSVP object header.

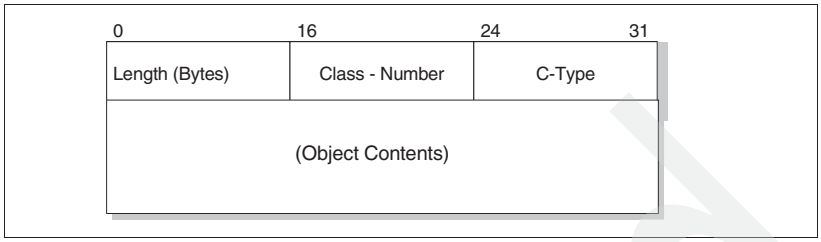


Figure 8-10 RSVP object header

Where:

Length 16-bit field that contains the object length in bytes. This must be a multiple of 4. The minimum length is 4 bytes.

Class-Number Identifies the object class. The following classes are defined:

NULL The NULL object has a Class-Number of zero. The length of this object must be at least 4, but can be any multiple of 4. The NULL object can appear anywhere in the object sequence of an RSVP message. The content is ignored by the receiver.

Session The Session object contains the IP destination address, the IP protocol ID, and the destination port to define a specific session for the other objects that follow. The Session object is required in every RSVP message.

RSVP_HOP The RSVP_HOP object contains the IP address of the node that sent this message and a logical outgoing interface handle. For downstream messages (for example, path messages), the RSVP_HOP object represents a PHOP (previous hop) object, and for upstream messages (for example, Resv messages), it represents an NHOP (next hop) object.

Time_Values The Time_Values object contains the refresh period for path and reservation messages. If these messages are not refreshed within the specified time period, the path or reservation state is canceled.

Archived

Style	The Style object defines the reservation style and some style-specific information that is not in Flowspec or Filterspec. The Style object is required in every Resv message.
Flowspec	This object specifies the required QoS in reservation messages.
Filterspec	The Filterspec object defines which data packets receive the QoS specified in the Flowspec.
Sender_Template	This object contains the sender IP address and additional demultiplexing information, which is used to identify a sender. The Sender_Template is required in every Path message.
Sender_Tspec	This object defines the traffic characteristics of a data flow from a sender. The Sender_Tspec is required in all path messages.
Adspec	The Adspec object is used to provide advertising information to the traffic control modules in the RSVP nodes along the path.
Error_Spec	This object specifies an error in a PathErr or ResvErr, or a confirmation in a ResvConf message.
Policy_Data	This object contains information that allows a policy module to decide whether an associated reservation is administratively permitted or not. It can be used in Path, Resv, PathErr, or ResvErr messages.
Integrity	The Integrity object contains cryptographic data to authenticate the originating node and to verify the contents of an RSVP message.
Scope	The Scope object contains an explicit list of sender hosts to which the information in the message are sent. The object can appear in a Resv, ResvErr, or ResvTear message.
Resv_Confirm	This object contains the IP address of a receiver that requests confirmation for its reservation. It can be used in a Resv or ResvConf message.
C-Type	The C-Type specifies the object type within the class number. Different object types are used for IPv4 and IPv6.
Object contents	The object content depends on the object type and has a maximum length of 65528 bytes.

All RSVP messages are built of a variable number of objects. The recommended object order for the most important RSVP messages, the path, and the Resv message, are shown in Figure 8-11, which gives an overview of the format of the RSVP path message. Objects that can appear in a path message, but that are not required, are in parentheses.

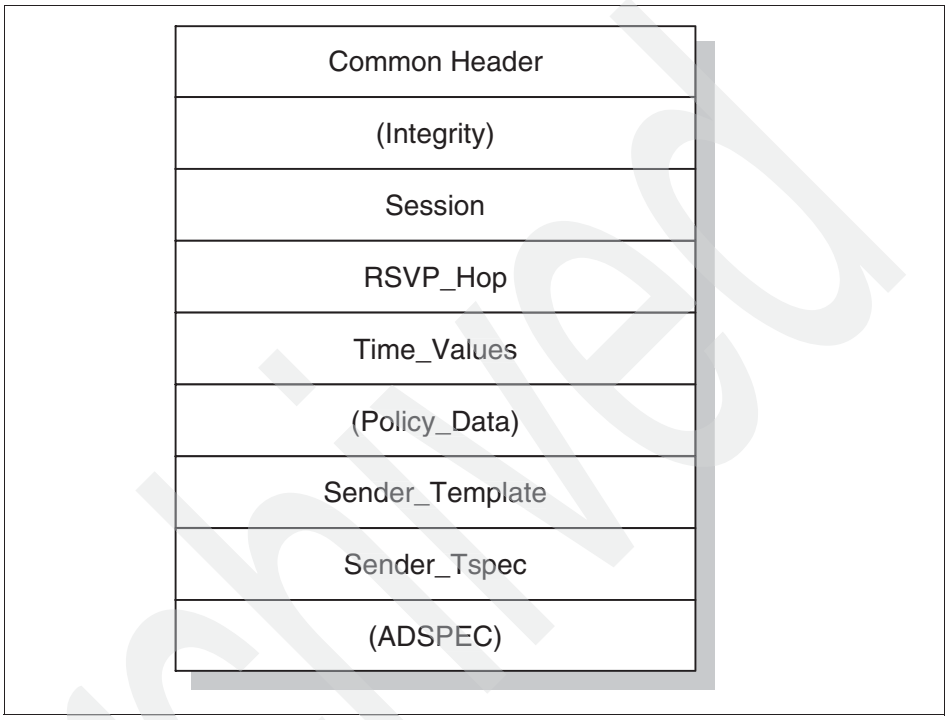


Figure 8-11 RSVP path message format

If the Integrity object is used in the path message, it must immediately follow the common header. The order of the other objects can differ in different RSVP implementations, but the order shown in Figure 8-11 is recommended by the RFC.

Archived

The RSVP Resv messages looks similar to the path message. Figure 8-12 shows the objects used for reservation messages.

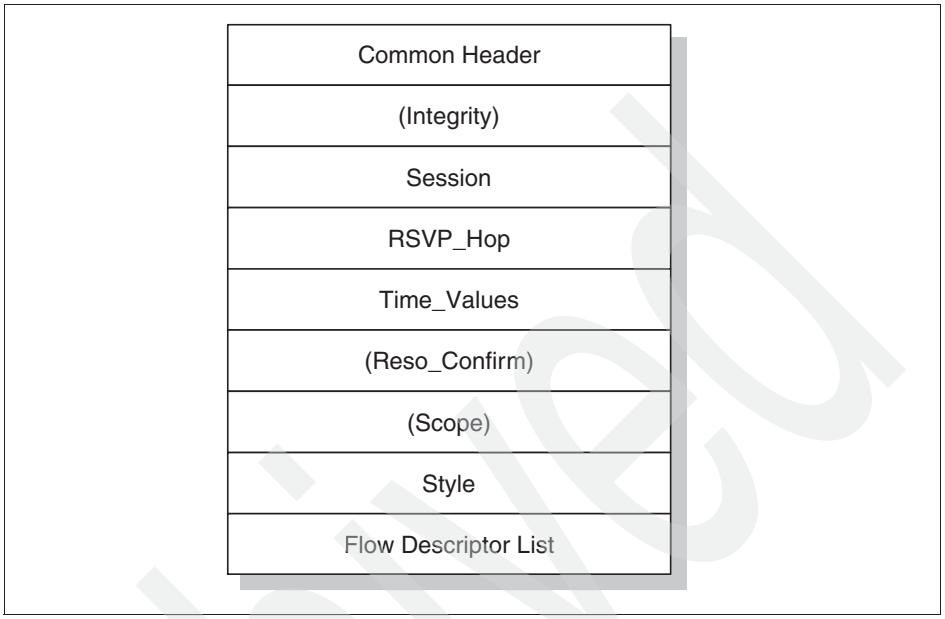


Figure 8-12 RSVP Resv message format

As in the path message, the Integrity object must follow the common header if it is used. Another restriction applies for the Style object and the following flow descriptor list. They must occur at the end of the message. The order of the other objects follows the recommendation from the RFC.

For a detailed description of the RSVP message structure and the handling of the different reservation styles in reservation messages, consult RFC 2205.

8.2.5 Integrated Services outlook

Integrated Services is designed to provide end-to-end quality of service (QoS) to applications over heterogeneous networks. This means that Integrated Services has to be supported by several different network types and devices. It also means that elements within the network, such as routers, need information to provide the requested service for an end-to-end QoS flow. This information setup in routers is done by the Resource Reservation Protocol (RSVP). RSVP is a signaling protocol that can carry Integrated Services information.

Although RSVP can be used to request resources from the network, Integrated Services defines the needed service types, quantifying resource requirements and determining the availability of the requested resource.

There are some factors that have prevented the deployment of RSVP and, thus, Integrated Services in the Internet. These include:

- Only a small number of hosts currently generate RSVP signalling. Although the number is expected to grow in the near future, many applications cannot generate RSVP signalling.
- Integrated Services is based on flow-state and flow-processing. If flow-processing rises dramatically, it might become a scalabiltiy concern for large networks.
- The necessary policy control mechanisms, such as access control authentication and accounting, have only recently become available.

The requirements of the market will determine if Integrated Services with RSVP will inspire service providers to use these protocols. But this requires that network devices (for example, routers) need the required software support.

Another aspect also needs to be considered. Support of Integrated Services running over Differentiated Services networks is a possibility. This solution provides:

- End-to-end QoS for applications, such as IP telephony and video on demand.
- Intserv enables hosts to request per-flow and quantify able resources along the end-to-end path, including feedback about admission to the resources.
- Diffserv eliminates the need for per-flow state and per-flow processing, and therefore enables the scalability across large networks.

8.3 Differentiated Services

The Differentiated Services (DS) concept is currently under development at the IETF DS working group. The DS specifications are defined in some IETF Internet drafts. There is no RFC available yet. This section gives an overview of the basics and ideas to provide service differentiation in the Internet. Because the concept is still under development, some of the specifications mentioned in this book might be changed in the final definition of differentiated services.

The goal of DS development is to provide differentiated classes of service for Internet traffic to support various types of applications and meet specific business requirements. DS offers predictable performance (delay, throughput, packet loss, and so on) for a given load at a given time. The difference between

Integrated Services, described in 8.2.5, “Integrated Services outlook” on page 308 and Differentiated Services is that DS provides scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. It is not necessary to perform a unique QoS reservation for each flow. With DS, the Internet traffic is split into different classes with different QoS requirements.

A central component of DS is the service level agreement (SLA). An SLA is a service contract between a client and a service provider that specifies the details of the traffic classifying and the corresponding forwarding service a client should receive. A client can be a user organization or another DS domain. The service provider must assure that the traffic of a client, with whom it has an SLA, gets the contracted QoS. Therefore, the service provider's network administration must set up the appropriate service policies and measure the network performance to guarantee the agreed traffic performance.

To distinguish the data packets from different clients in DS-capable network devices, the IP packets are modified in a specific field. A small bit-pattern, called the *DS field*, in each IP packet is used to mark the packets that receive a particular forwarding treatment at each network node. The DS field uses the space of the former TOS octet in the IPv4 IP header and the traffic class octet in the IPv6 header. All network traffic inside of a domain receives a service that depends on the traffic class that is specified in the DS field.

To provide SLA conform services, the following mechanisms must be combined in a network:

- Setting bits in the DS field (TOS octet) at network edges and administrative boundaries.
- Using those bits to determine how packets are treated by the routers inside the network.
- Conditioning the marked packets at network boundaries in accordance with the QoS requirements of each service.

The currently defined DS architecture only provides service differentiation in one direction and is therefore asymmetric. Development of a complementary symmetric architecture is a topic of current research. The following section describes the DS architecture in more detail.

8.3.1 Differentiated Services architecture

Unlike Integrated Services, QoS guarantees made with Differentiated Services are static and stay long-term in routers. This means that applications using DS do not need to set up QoS reservations for specific data packets. All traffic that

passes DS-capable networks can receive a specific QoS. The data packets must be marked with the DS field that is interpreted by the routers in the network.

Per-hop behavior (PHB)

The DS field uses six bits to determine the Differentiated Services Code Point (DSCP). This code point will be used by each node in the net to select the PHB. A two-bit currently unused (CU) field is reserved. The value of the CU bits are ignored by Differentiated Services-compliant nodes when PHP is used for received packets. Figure 8-13 shows the structure of the defined DS field.

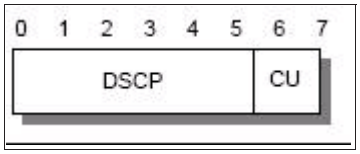


Figure 8-13 DS field

Each DS-capable network device must have information about how packets with different DS field should be handled. In the DS specifications, this information is called the *per-hop behavior* (PHB). It is a description of the forwarding treatment a packet receives at a given network node. The DSCP value in the DS field is used to select the PHB a packet experiences at each node. To provide predictable services, per-hop behaviors need to be available in all routers in a Differentiated Services-capable network. The PHB can be described as a set of parameters inside of a router that can be used to control how packets are scheduled onto an output interface. This can be a number of separate queues with priorities that can be set, parameters for queue lengths, or drop algorithms and drop preference weights for packets.

DS requires routers that support queue scheduling and management to prioritize outbound packets and control the queue depth to minimize congestion in the network. The traditional FIFO queuing in common Internet routers provides no service differentiation and can lead to network performance problems. The packet treatment inside of a router depends on the router's capabilities and its particular configuration, and it is selected by the DS field in the IP packet. For example, if a IP packet reaches a router with eight different queues that all have

different priorities, the DS field can be used to select which queue is liable for the routing of this packet. The scale reaches from zero, for lowest priority, to seven for highest priority. See Figure 8-14 as an example.

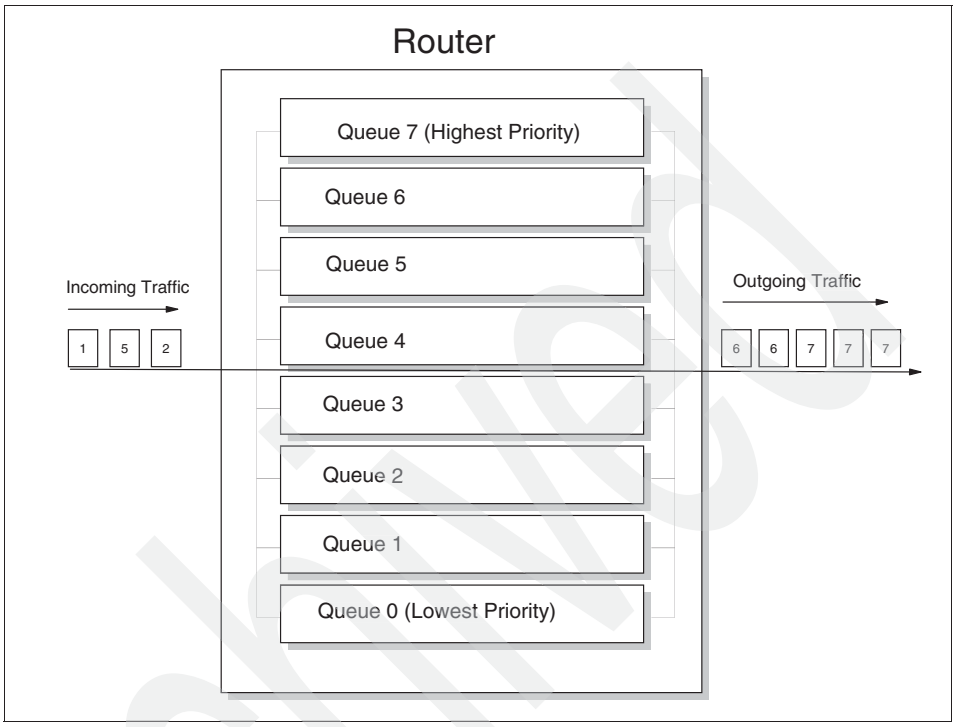


Figure 8-14 DS routing example

Another example is a router that has a single queue with multiple drop priorities for data packets. It uses the DS field to select the drop preference for the packets in the queue. A value of zero means “it is most likely to drop this packet,” and seven means “it is least likely to drop this packet.” Another possible constellation is four queues with two levels of drop preference in each.

To make sure that the per-hop behaviors in each router are functionally equivalent, certain common PHBs must be defined in future DS specifications to avoid having the same DS field value causing different forwarding behaviors in different routers of one DS domain. This means that in future DS specifications, some unique PHB values must be defined that represent specific service classes. All routers in one DS domain must know which service a packet with a specific PHB should receive. The DiffServ Working Group will propose PHBs that should be used to provide differentiated services. Some of these proposed PHBs will be standardized; others might have widespread use, and still others might remain experimental.

PHBs will be defined in groups. A PHB group is a a set of one or more PHBs that can only be specified and implemented simultaneously because of queue servicing or queue management policies that apply to all PHBs in one group. A default PHB must be available in all DS-compliant nodes. It represents the standard best-effort forwarding behavior available in existing routers. When no other agreements are in place, it is assumed that packets belong to this service level. The IETF working group recommends that you use the DSCP value 000000 in the DS field to define the default PHB.

Another PHB that is proposed for standardization is the *Expedited Forwarding* (EF) PHB. It is a high priority behavior that is typically used for network control traffic, such as routing updates. The value 101100 in the DSCP field of the DS field is recommended for the EF PHB.

8.3.2 Organization of the DSCP

There are some IANA considerations concerning the DSCP. The codepoint space for the DSCP distinguishes between 64 codepoint values. The proposal is to divide the space into tree pools. Pool1 can be used for standard actions. The other pools can be used for experimental local usage, where one of the two pools is provided for experimental local use in the near future.

The proposal is to divide the space into three pools (see Table 8-1).

Table 8-1 DSCP pools

Pool	Codepoint space	Assignment
1	xxxxx0	Standard action
2	xxxx11	Experimental/local use
3	xxxx01	Future experimental /local use

Differentiated Services domains

The setup of QoS guarantees is not made for specific end-to-end connections but for well-defined Differentiated Services domains. The IETF working group defines a Differentiated Services domain as a contiguous portion of the Internet over which a consistent set of Differentiated Services policies are administered in a coordinated fashion. It can represent different administrative domains or autonomous systems, different trust regions, and different network technologies, such as cell or frame-based techniques, hosts, and routers. A DS domain consists of boundary components that are used to connect different DS domains to each other and interior components that are only used inside of the domains.

Archived

Figure 8-15 shows the use of boundary and interior components for two DS domains.

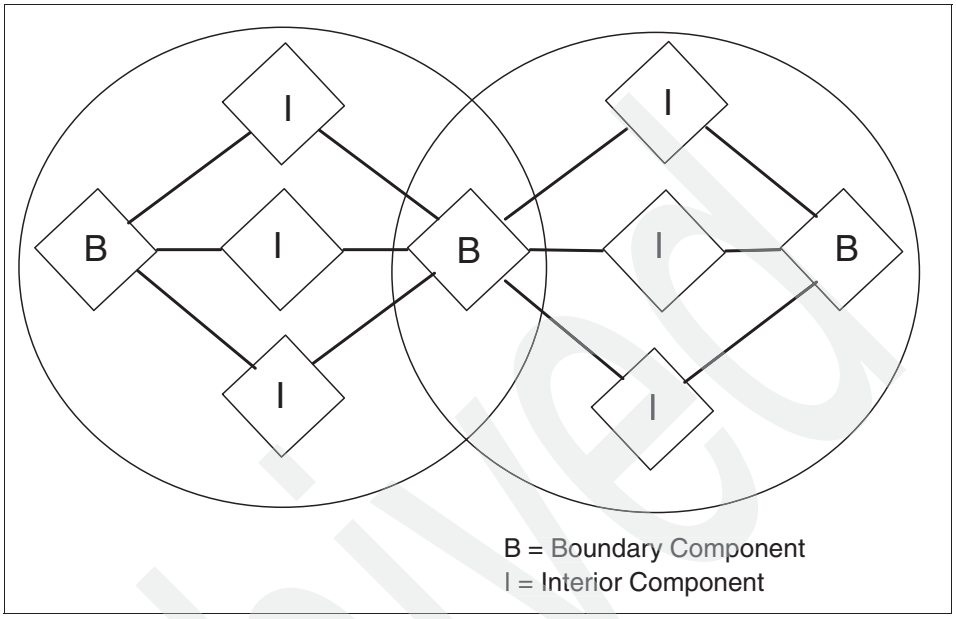


Figure 8-15 DS domain

A DS domain normally consists of one or more networks under the same administration. This can be, for example, a corporate intranet or an Internet service provider (ISP). The administrator of the DS domain is responsible for ensuring that adequate resources are provisioned and reserved to support the SLAs offered by the domain. Network administrators must use appropriate measurement techniques to monitor if the network resources in a DS domain are sufficient to satisfy all authorized QoS requests.

DS boundary nodes

All data packets that travel from one DS domain to another must pass a boundary node, which can be a router, a host, or a firewall. A DS boundary node that handles traffic leaving a DS domain is called an *egress node* and a boundary node that handles traffic entering a DS domain is called an *ingress node*. Normally, DS boundary nodes act both as an ingress node and an egress node, depending on the traffic direction. The ingress node must make sure that the packets entering a domain receive the same QoS as in the domain the packets traveled before. A DS egress node performs conditioning functions on traffic that is forwarded to a directly connected peering domain. The traffic conditioning is done inside of a boundary node by a *traffic conditioner*. It classifies, marks, and

possibly conditions packets that enter or leave the DS domain. A traffic conditioner consists of the following components:

Classifier A classifier selects packets based on their packet header and forwards the packets that match the classifier rules for further processing. The DS model specifies two types of packet classifiers:

- Multi-field (MF) classifiers, which can classify on the DS field as well as on any other IP header field, for example, the IP address and the port number, like an RSVP classifier
- Behavior Aggregate (BA) classifiers, which only classify on the bits in the DS field

Meter Traffic meters measure if the forwarding of the packets that are selected by the classifier corresponds to the traffic profile that describes the QoS for the SLA between client and service provider. A meter passes state information to other conditioning functions to trigger a particular action for each packet, which either does or does not comply with the requested QoS requirements.

Marker DS markers set the DS field of the incoming IP packets to a particular bit pattern. The PHB is set in the first 6 bits of the DS field so that the marked packets are forwarded inside of the DS domain according to the SLA between service provider and client.

Shaper/dropper Packet shapers and droppers cause conformance to some configured traffic properties, for example, a token bucket filter, as described in 8.2.1, “Service classes” on page 292. They use different methods to bring the stream into compliance with a traffic profile. Shapers delay some or all of the packets. A shaper usually has a finite-size buffer, and packets can be discarded if there is not sufficient buffer space to hold the delayed packets. Droppers discard some or all of the packets. This process is know as *policing* the stream. A dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero packets.

Archive

The traffic conditioner is mainly used in DS boundary components, but it can also be implemented in an interior component. Figure 8-16 shows the cooperation of the traffic conditioner components.

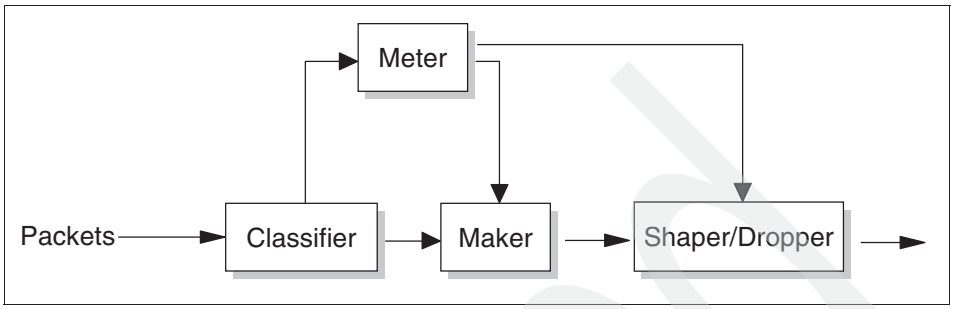


Figure 8-16 DS traffic conditioner

The traffic conditioner in a boundary component makes sure that packets that transit the domain are correctly marked to select a PHB from one of the PHB groups supported within the domain. This is necessary because different DS domains can have different groups of PHBs, which means that the same entry in the DS field can be interpreted variably in different domains.

For example, in the first domain a packet traverses, all routers have four queues with different queue priorities (0-3). Packets with a PHB value of three are routed with the highest priority. But in the next domain the packet travels through, all routers have eight different queues and all packets with the PHB value of seven are routed with the highest priority. The packet that was forwarded in the first domain with high priority has only medium priority in the second domain. This might violate the SLA contract between the client and the service provider. Therefore, the traffic conditioner in the boundary router that connects the two domains must assure that the PHB value is remarked from three to seven if the packet travels from the first to the second domain.

Figure 8-17 shows an example of remarking data packets that travel trough two different domains.

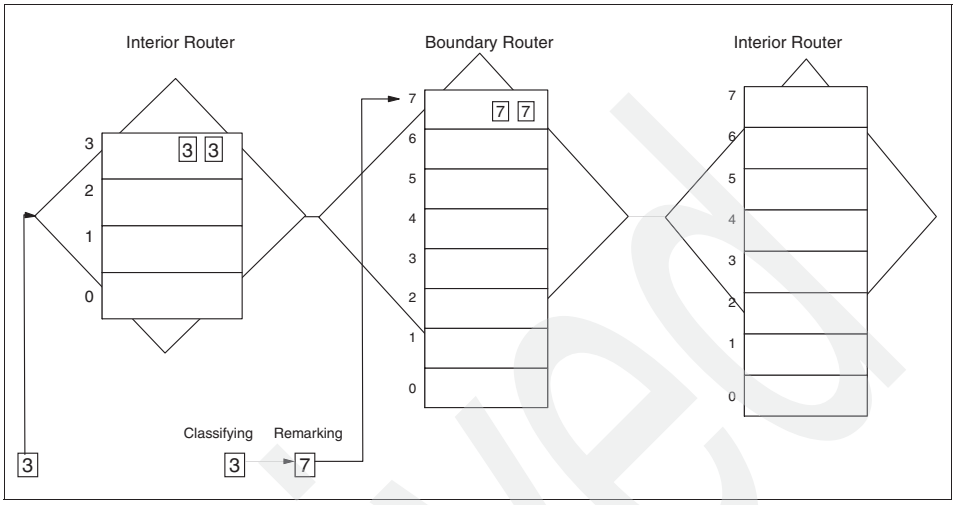


Figure 8-17 Remarking data packets

If a data packet travels through multiple domains, the DS field can be remarked at every boundary component to guarantee the QoS that was contracted in the SLA. The SLA contains the details of the *Traffic Conditioning Agreement (TCA)* that specifies classifier rules and temporal properties of a traffic stream. The TCA contains information about how the metering, marking, discarding, and shaping of packets must be done in the traffic conditioner to fulfill the SLA. The TCA information must be available in all boundary components of a DS network to guarantee that packets passing through different DS domains receive the same service in each domain.

DS interior components

The interior components of a DS domain select the forwarding behavior for packets based on their DS field. The interior component is usually a router that contains a traffic prioritization algorithm. Because the value of the DS field normally does not change inside of a DS domain, all interior routers must use the same traffic forwarding policies to comply with the QoS agreement. Data packets with different PHB values in the DS field receive different QoSs according to the QoS definitions for this PHB. Because all interior routers in a domain use the same policy functions for incoming traffic, the traffic conditioning inside of an interior node is done only by a packet classifier. It selects packets based on their PHB value or other IP header fields and forwards the packets to the queue management and scheduling instance of the node.

Figure 8-18 shows the traffic conditioning in an interior node.

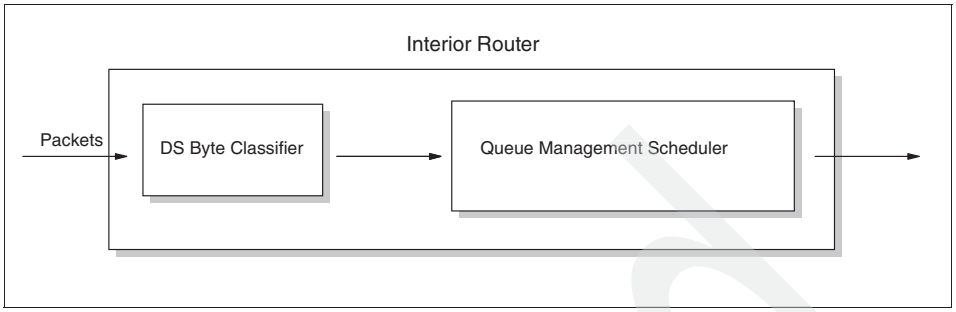


Figure 8-18 DS interior component

Traffic classifying and prioritized routing is done in every interior component of a DS domain. After a data packet has crossed a domain, it reaches the boundary router of the next domain and possibly gets remarked to cross this domain with the requested QoS.

Source domains

The IETF DS working group defines a source domain as the domain that contains one or more nodes that originate the traffic that receives a particular service. Traffic sources and intermediate nodes within a source domain can perform traffic classification and conditioning functions. The traffic that is sent from a source domain can be marked by the traffic sources directly or by intermediate nodes before leaving the source domain.

In this context, it is important to understand that the first PHB marking of the data packets is not done by the sending application itself. Applications do not notice the availability of Differentiated Services in a network. Therefore, applications using DS networks must not be rewritten to support DS. This is an important difference from Integrated Services, where most applications support the RSVP protocol directly when some code changes are necessary.

The first PHB marking of packets that are sent from an application is done in the source host or the first router the packet passes. The packets are identified with their IP address and source port. For example, a client has an SLA with a service provider that guarantees a higher priority for the packets sent by an audio application. The audio application sends the data packets through a specific port and can be recognized in multi-field classifiers. This classifier type recognizes the IP address and port number of a packet and can distinguish the packets from different applications. If the host contains a traffic conditioner with an MF classifier, the IP packet can be marked with the appropriate PHB value and consequently receives the QoSs that are requested by the client. If the host does not contain a traffic conditioner, the initial marking of the packets is done by the

first router in the source domain that supports traffic conditioning. Figure 8-19 shows the initial marking of a packet inside of a host and a router.

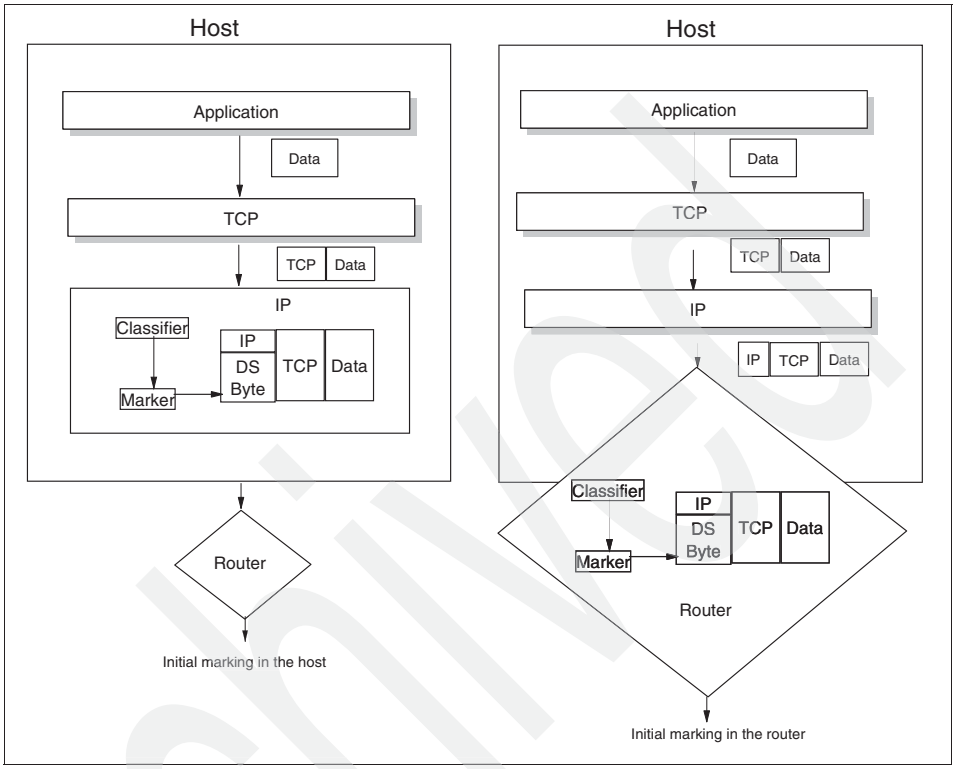


Figure 8-19 Initial marking of data packets

In our example, the DS network has the policy that the packets from the audio application should have higher priority than other packets. The sender host can mark the DS field of all outgoing packets with a DS codepoint that indicates higher priority. Alternatively, the first-hop router directly connected to the sender's host can classify the traffic and mark the packets with the correct DS codepoint. The source DS domain is responsible for ensuring that the aggregated traffic toward its provider DS domain conforms to the SLA between client and service provider. The boundary node of the source domain should also monitor that the provided service conforms to the requested service and can police, shape, or re-mark packets as necessary.

Integrated Services (Intserv) over Diffserv networks

The basic idea is to use both architectures to provide an end-to-end, quantitative QoS, which will also allow scalability. This will be achieved by applying the

Intserv model end-to-end across a network containing one or more Diffserv regions.

Intserv views the Diffserv regions as virtual links connecting Intserv-capable routers or hosts running Intserv. Within the Diffserv regions, the routers are implemented with specific PHB definitions to provide aggregate traffic control. The total amount of traffic that is admitted into the Diffserv region may be limited by a determined policy at the edges of the Diffserv network. The Intserv traffic has to be adapted to the limits of the Diffserv region.

There are two possible approaches for connecting Intserv networks with Diffserv networks:

- Resources within the Diffserv network or region include RSVP-aware devices that participate in RSVP signalling.
- Resources within the Diffserv region include no RSVP signalling.

In our sample (Figure 8-20), we describe the second configuration, because it is assumed to become the most current approach to use.

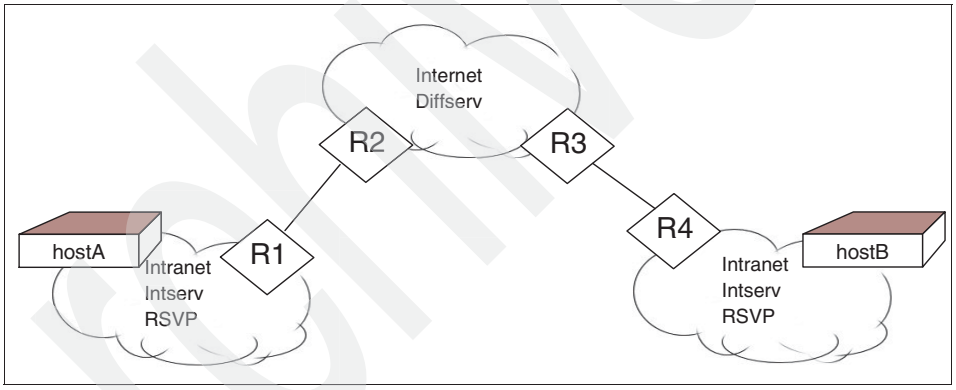


Figure 8-20 Integrated Services and RSVP using Differentiated Services

This configuration consists of two RSVP-capable intranets that are connected to Diffserv regions. Within the intranets, there are hosts using RSVP to communicate the quantitative QoS requirements with a partner host in another intranet running QoS-aware applications.

The intranets contain edge routers R1 and R4, which are adjacent to the Diffserv region interface. They are connected with the border routers R2 and R3, which are located within the Diffserv region.

The RSVP signalling process is initiated by the service requesting application on the host (for example, hostA). Traffic control on the host can mark the DSCP in

the transmitted packets and shape transmitted traffic to the requirements of the Intserv services in use.

The RSVP end-to-end signaling messages are exchanged between the hosts in the intranets. Thus, the RSVP/Intserv resource reservation is accomplished outside the Diffserv region.

The edge routers act as admission control agents of the Diffserv network. They process the signaling messages from the hosts in both intranets and apply admission control. Admission control is based on resources available within the Diffserv region. It is also defined in a company's policy for the intranets.

Because the border routers in our sample (R2 and R3) are not aware of RSVP, they act as pure Diffserv routers. These routers control and submit packets based on the specified DSCP and the agreement for the host's aggregate traffic control.

The Diffserv network or region supports aggregate traffic control and is assumed to be incapable of MF classification. Therefore, any RSVP messages will pass transparently with negligible performance impact through a Diffserv network region.

The next aspect to be considered is the mapping of the Intserv service type and the set of quantitative parameters, known as flowspec. Because Diffserv networks use PHB or a set of PHBs, a mapping of the Intserv flowspec has to be defined accordingly. The mapping value is a bit combination in the DSCP. However, this mapping has to viewed under bandwidth management considerations for the Diffserv network.

The DSCP value has been made known to all routers in the Diffserv network. The question arises of how the DSCP will be propagated to these routers.

There are two choices:

- DSCPs can be marked at the entrance of the Diffserv region (at the boundary routers). In this case, they can be also re-marked at the exit of the Diffserv region (at the other boundary router).
- DSCP marking can occur in a host or in a router of the intranet. In this case, the appropriate mapping needs to be communicated to the marking device. This can be provided by RSVP.

The following sequence shows how an application obtains end-to-end QoS support:

1. HostA, attached to an intranet, requests a service from hostB, attached to another intranet. Both intranets are connected by a Diffserv network (see Figure 8-20 on page 320).

2. HostA generates a RSVP PATH message, which describes the traffic offered by the sending application.
3. The PATH message is sent over the intranet to router R1. Standard RSVP/Intserv processing is done by the network devices within the intranet.
4. The PATH state is defined in the router R1, and is forwarded to the Router R2 in the Diffserv network.
5. The PATH message is ignored in the Diffserv network at R2 and R3. It is sent to R4 in the intranet and to hostB.
6. When the PATH message is received by hostB, a RSVP RESV message is built, indicating the offered traffic of a specific Intserv service type.
7. The RESV message is sent to the Diffserv network.
8. The Diffserv network transparently transmits the message to router R1.
9. In R1, the RESV message triggers admission control processing. This means requested resources in the initial RSVP/Intserv request are compared to the resources available in the Diffserv network at the corresponding Diffserv service level. The corresponding service level is determined by the Intserv to Diffserv mapping function. The availability of resources is determined by the capacity defined in the SLA.
10. If R1 approves the request, the RESV message is admitted and is allowed to be sent to the sender, hostA. R1 updates its tables with reduced capacity available at the admitted service level on this particular transmit interface.
11. If the RESV message is not rejected by any RSVP node in the intranet, it will be received at hostA. The QoS process interprets the receipt of the message as an indication that the specified message flow has been admitted for the specified Intserv service type. It also learns the DSCP marking, which will be used for subsequent packets to be sent for this flow.

8.3.3 Configuration and administration of DS with LDAP

In a Differentiated Services network, the service level information must be provided to all network elements to ensure correct administrative control of bandwidth, delay, or dropping preferences for a given customer flow. All DS boundary components must have the same policy information for the defined service levels. This makes sure that the packets marked with the DS field receive the same service in all DS domains. If only one domain in the DS network has different policy information, it is possible that the data packets passing this domain will not receive the service that was contracted in the SLA between client and service provider.

Network administrators can define different service levels for different clients and manually provide this information to all boundary components. This policy

information remains statically in the network components until the next manual change.

But in dynamic network environments, it is necessary to enable flexible definitions of class-based packet handling behaviors and class-based policy control. Administrative policies can change in a running environment, so it is necessary to store the policies in a directory-based repository. The policy information from the directory can be distributed across multiple physical servers, but the administration is done for a single entity by the network administrator. The directory information must be propagated on all network elements, such as hosts, proxies, and routers, that use the policy information for traffic conditioning in the DS network. In today's heterogeneous environments, it is likely that network devices and administrative tools are developed by different vendors. Therefore, it is necessary to use a standardized format to store the administrative policies in the directory server function and a standardized mechanism to provide the directory information to the DS boundary components, which act as directory clients. These functions are provided by the *Lightweight Directory Access Protocol (LDAP)*, which is a commonly used industry standard for directory accessing. LDAP is a widely deployed and simple protocol for directory access. Policy rules for different service levels are stored in directories as LDAP schemas and can be downloaded to devices that implement the policies, such as hosts, routers, policy servers, or proxies.

Archive

Figure 8-21 shows the cooperation of the DS network elements with the LDAP server.

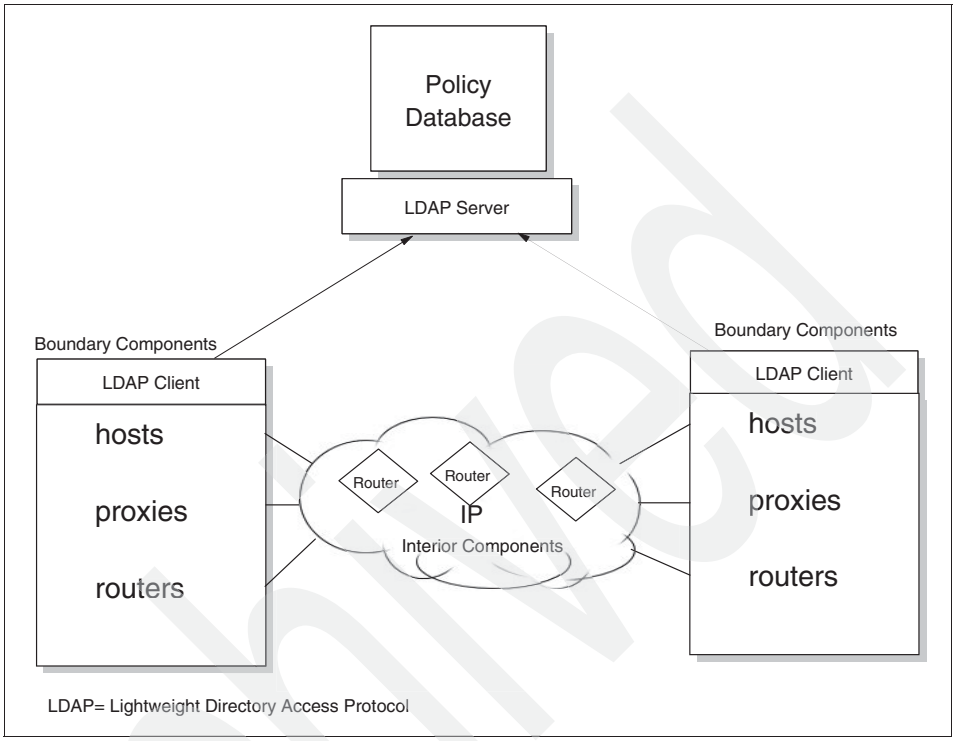


Figure 8-21 Administration of DS components with LDAP

Using Differentiated Services with IPSec

The IPSec protocol (described in 22.4.3, “Encapsulating Security Payload (ESP)” on page 817) does not use the DS field in an IP header for its cryptographic calculations. Therefore, modification of the DS field by a network node has no effect on IPSec's end-to-end security, because it cannot cause any IPSec integrity check to fail. This makes it possible to use IPSec-secured packets in DS networks.

IPSec's tunnel mode provides security for the encapsulated IP header's DS field. A tunnel mode IPSec packet contains an outer header that is supplied by the tunnel start point and an encapsulated inner header that is supplied by the host that originally sent the packet.

Processing of the DS field in the presence of IPSec tunnels then works as follows:

1. The node where the IPSec tunnel begins encapsulates the incoming IP packets with an outer IP header and sets the DS field of the outer header accordingly to the SLA in the local DS domain.
2. The secured packet travels through the DS network, and intermediate nodes modify the DS field in the outer IP header, as appropriate.
3. If a packet reaches the end of an IPSec tunnel, the outer IP header is stripped off by the tunnel end node and the packet is forwarded using the information contained in the inner (original) IP header.
4. If the DS domain of the original datagram is different from the DS domain where the IPSec tunnel ends, the tunnel end node must modify the DS field of the inner header to match the SLA in its domain. The tunnel end node then effectively acts as a DS ingress node.
5. As the packet travels in the DS network on the other side of the IPSec tunnel, intermediate nodes use the original IP header to modify the DS field.

8.4 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 1349 – Type of Service in the Internet Protocol Suite (July 1992)
- RFC 1633 – Integrated Services in the Internet Architecture: An Overview (June 1994)
- RFC 4495 – A Resource Reservation Protocol (RSVP) Extension for the Reduction of Bandwidth of a Reservation Flow (May 2006)
- RFC 2206 – RSVP Management Information Base Using SMIv2 (September 1997)
- RFC 2207 – RSVP Extensions for IPSEC Data Flows (September 1997)
- RFC 2208 – Resource Reservation Protocol (RSVP) – Version 1 Applicability Statement (September 1997)
- RFC 2209 – Resource Reservation Protocol (RSVP) – Version 1 Message Processing Rules (September 1997)
- RFC 2210 – The Use of RSVP with IETF Integrated Services (September 1997)
- RFC 2211 – Specification of the Controlled Load Network Element Service (September 1997)

Archived

- RFC 2212 – Specification of Guaranteed Quality of Service (September 1997)
- RFC 2474 – Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (December 1998)

Archived

9

IP version 6

This chapter discusses the concepts and architecture of IP version 6. This chapter includes the following topics:

- An overview of IPv6 new features
- An examination of the IPv6 packet format
- An explanation of additional IPv6 functions
- A description of associated protocols
- A review of IPv6 mobility applications
- A projection of new opportunities opened up by IPv6
- An explanation of the needs for next generation IPv6
- An exploration into IPv4-to-IPv6 transition paths

© Copyright IBM Corp. 1989-2006. All rights reserved.

9.1 IPv6 introduction

Internet Protocol version 6 (IPv6) is the replacement for version 4 (IPv4). This section discusses the expanded address space capabilities of IPv6 and includes a brief feature overview.

9.1.1 IP growth

The Internet is growing extremely rapidly. The latest Internet Domain Survey[1], conducted in January 2006, counted 395 million hosts (374,991,609 hosts to be exact).

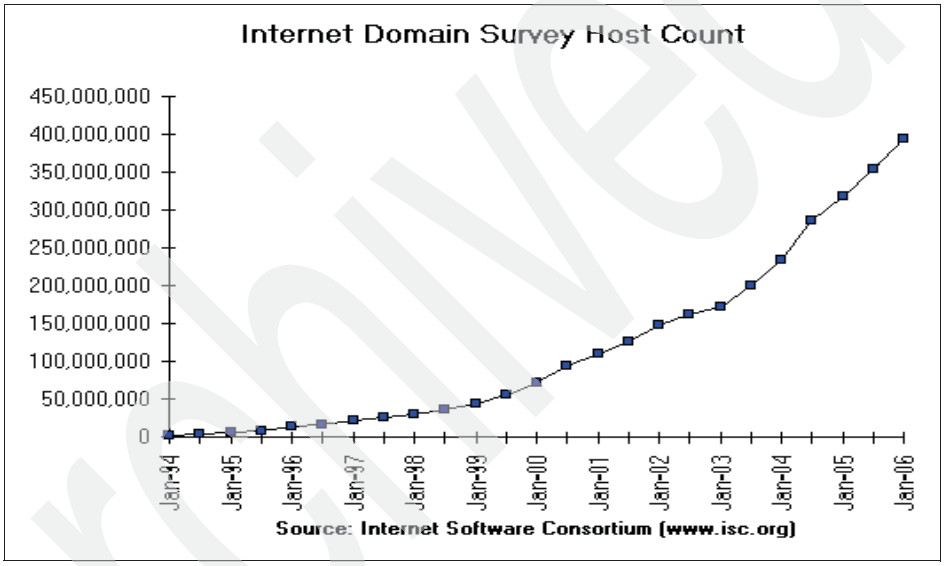


Figure 9-1 The total number of IP hosts growth

The IPv4 addressing scheme, with a 32-bit address field, provides for over 4 billion possible addresses, so it might seem more than adequate to the task of addressing all of the hosts on the Internet, since there appears to be room to accommodate 40 times as many Internet hosts. Unfortunately, this is not the case for a number of reasons, including the following:

- An IP address is divided into a network portion and a local portion which are administered separately. Although the address space within a network may be very sparsely filled, allocating a portion of the address space (range of IP addresses) to a particular administrative domain makes all addresses within that range unavailable for allocation elsewhere.

[1] Source: Internet Software Consortium (http://www.isc.org/)

- The address space for networks is structured into Class A, B, and C networks of differing sizes, and the space within each needs to be considered separately.
- The IP addressing model requires that unique network numbers be assigned to all IP networks, whether or not they are actually connected to the Internet.
- It is anticipated that growth of TCP/IP usage into new areas outside the traditional connected PC will shortly result in a rapid explosion of demand for IP addresses. For example, widespread use of TCP/IP for interconnecting hand-held devices, electronic point-of-sale terminals or for Web-enabled television receivers (all devices that are now available) will enormously increase the number of IP hosts.

These factors mean that the address space is much more constrained than our simple analysis would indicate. This problem is called *IP Address Exhaustion*. Methods of relieving this problem are already being employed, but eventually, the present IP address space will be exhausted. The Internet Engineering Task Force (IETF) set up a working group on *Address Lifetime Expectations (ALE)* with the express purpose of providing estimates of when exhaustion of the IP will become an intractable problem. Their final estimates (reported in the ALE working group minutes for December 1994) were that the IP address space would be exhausted at some point between 2005 and 2011. Since then, their position may have changed somewhat, in that the use of Classless Inter Domain Routing (CIDR) and the increased use of Dynamic Host Configuration Protocol (DHCP) may have relieved pressure on the address space.

But on the other hand, current growth rates are probably exceeding that expectation. Apart from address exhaustion, other restrictions in IPv4 also call for the definition of a new IP protocol:

1. Even with the use of CIDR, routing tables, primarily in the IP backbone routers, are growing too large to be manageable.
2. Traffic priority, or class of service, is vaguely defined, scarcely used, and not at all enforced in IPv4, but highly desirable for modern real-time applications.
3. The number of mobile data applications and devices grow quickly, IPv4 has difficulty in managing forwarding addresses and in realizing visitor-location network authentication.
4. There is no direct security support in IPv4. Various open and proprietary security solutions cause interoperability concerns. As the internet becomes the fabric of every life in the new cyber space, security enhancement to the infrastructure should be placed in the basic IP protocol.

In view of these issues, the IETF established an IPng (IP next generation) working group to make recommendations for the IP Next Generation Protocol.

Eventually, the specification for Internet Protocol, Version 6 (IPv6) was produced in RFC2460 as the latest version.

9.1.2 IPv6 feature overview

IPv6 offers the following significant features:

- A dramatically larger address space, which is said to be sufficient for at least the next 30 years
- Globally unique and hierarchical addressing, based on prefixes rather than address classes, to keep routing tables small and backbone routing efficient
- A mechanism for the auto-configuration of network interfaces
- Support for encapsulation of itself and other protocols
- Class of service that distinguishes types of data
- Improved multicast routing support (in preference to broadcasting)
- Built-in authentication and encryption
- Transition methods to migrate from IPv4
- Compatibility methods to coexist and communicate with IPv4

Note: IPv6 uses the term *packet* rather than *datagram*. The meaning is the same, although the formats are different.

IPv6 uses the term *node* for any system running IPv6, that is, a host or a router. An IPv6 host is a node that does not forward IPv6 packets that are not explicitly addressed to it. A router is a node that forwards IP packets not addressed to it.

9.2 The IPv6 header format

The format of the IPv6 packet header has been simplified from its counterpart in IPv4. The length of the IPv6 header increases to 40 bytes (from 20 bytes) and contains two 16-byte addresses (source and destination), preceded by 8 bytes of control information, as shown in Figure 9-2 on page 331. The IPv4 header has two 4-byte addresses preceded by 12 bytes of control information and possibly followed by option data. The reduction of the control information and the elimination of options in the header for most IP packets are intended to optimize the processing time per packet in a router. The infrequently used fields that have been removed from the header are moved to optional extension headers when they are required.

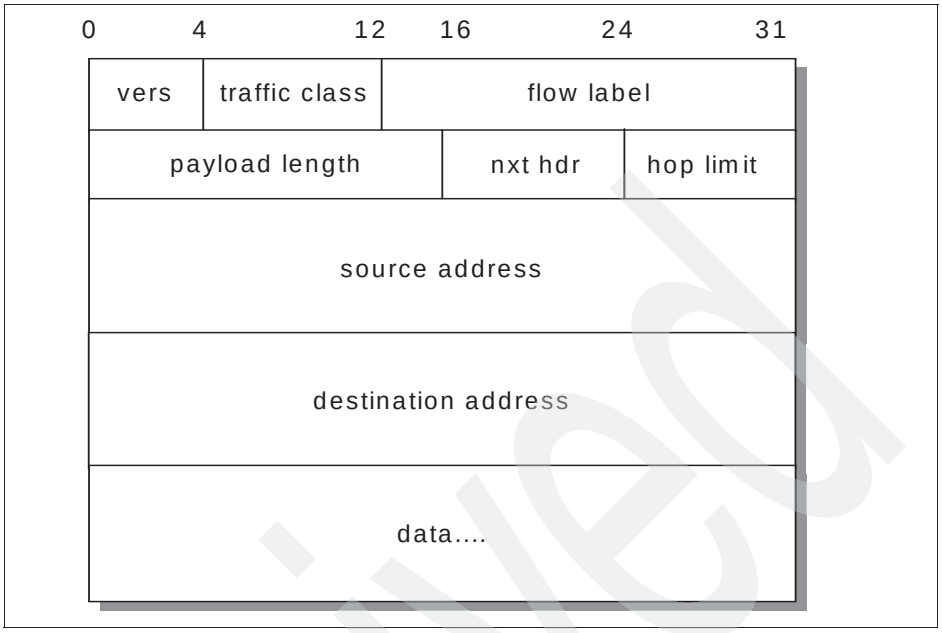


Figure 9-2 IP header format

Where:

Vers 4-bit Internet Protocol version number: 6.

Traffic class 8-bit traffic class value. See 9.2.3, “Traffic class” on page 345.

Flow label 20-bit field. See 9.2.4, “Flow labels” on page 346.

Payload length The length of the packet in bytes (excluding this header) encoded as a 16-bit unsigned integer. If length is greater than 64 KB, this field is 0 and an option header (Jumbo Payload) gives the true length.

Next header Indicates the type of header immediately following the basic IP header. It can indicate an IP option header or an upper layer protocol. The protocol numbers used are the same as those used in IPv4. The next header field is also used to indicate the presence of extension headers, which provide the mechanism for appending optional information to the IPv6 packet. The following values appear in IPv6 packets, in addition to those mentioned for IPv4:

41 Header

45 Interdomain Routing Protocol

Archived

46 Resource Reservation Protocol

58 IPv6 ICMP Packet

The following values are all extension headers:

0 Hop-by-Hop Options Header

43 IPv6 Routing Header

44 IPv6 Fragment Header

50 Encapsulating Security Payload

51 IPv6 Authentication Header

59 No Next Header

60 Destination Options Header

We discuss different types of extension headers in 9.2.1, “Extension headers” on page 333.

Hop limit This is similar to the IPv4 TTL field but it is now measured in hops and not seconds. It was changed for two reasons:

- IP normally forwards datagrams faster than one hop per second and the TTL field is always decremented on each hop, so, in practice, it is measured in hops and not seconds.
- Many IP implementations do not expire outstanding datagrams on the basis of elapsed time.

The packet is discarded after the hop limit is decremented to zero.

Source address A 128-bit address. We discuss IPv6 addresses in 9.2.2, “IPv6 addressing” on page 339.

Destination address A 128-bit address. We discuss IPv6 addresses in 9.2.2, “IPv6 addressing” on page 339.

A comparison of the IPv4 and IPv6 header formats shows that a number of IPv4 header fields have no direct equivalents in the IPv6 header:

- Type of service

  Type of service issues in IPv6 are handled using the *flow* concept, described in 9.2.4, “Flow labels” on page 346.

Archived

- Identification, fragmentation flags, and fragment offset

  Fragmented packets have an extension header rather than fragmentation information in the IPv6 header. This reduces the size of the basic IPv6 header, because higher-level protocols, particularly TCP, tend to avoid fragmentation of datagrams (this reduces the IPv6 header processing costs for the normal case). As noted later, IPv6 does not fragment packets en route to their destinations, only at the source.

- Header checksum

  Because transport protocols implement checksums, and because IPv6 includes an optional authentication header that can also be used to ensure integrity, IPv6 does *not* provide checksum monitoring of IP packets.

  Both TCP and UDP include a pseudo IP header in the checksums they use, so in these cases, the IP header in IPv4 is being checked twice.

  TCP and UDP, and any other protocols using the same checksum mechanisms running over IPv6, will continue to use a pseudo IP header although, obviously, the format of the pseudo IPv6 header will be different from the pseudo IPv4 header. ICMP, IGMP, and any other protocols that do not use a pseudo IP header over IPv4 use a pseudo IPv6 header in their checksums.

- Options

  All optional values associated with IPv6 packets are contained in extension headers, ensuring that the basic IP header is always the same size.

9.2.1 Extension headers

Every IPv6 packet starts with the basic header. In most cases, this will be the only header necessary to deliver the packet. Sometimes, however, it is necessary for additional information to be conveyed along with the packet to the destination or to intermediate systems on route (information that would previously have been carried in the Options field in an IPv4 datagram). Extension headers are used for this purpose.

Extension headers are placed immediately after the IPv6 basic packet header and are counted as part of the payload length. Each extension header (with the exception of 59) has its own 8-bit *Next Header field* as the first byte of the header that identifies the type of the following header. This structure allows IPv6 to chain multiple extension headers together. Figure 9-3 on page 334 shows an example packet with multiple extension headers.

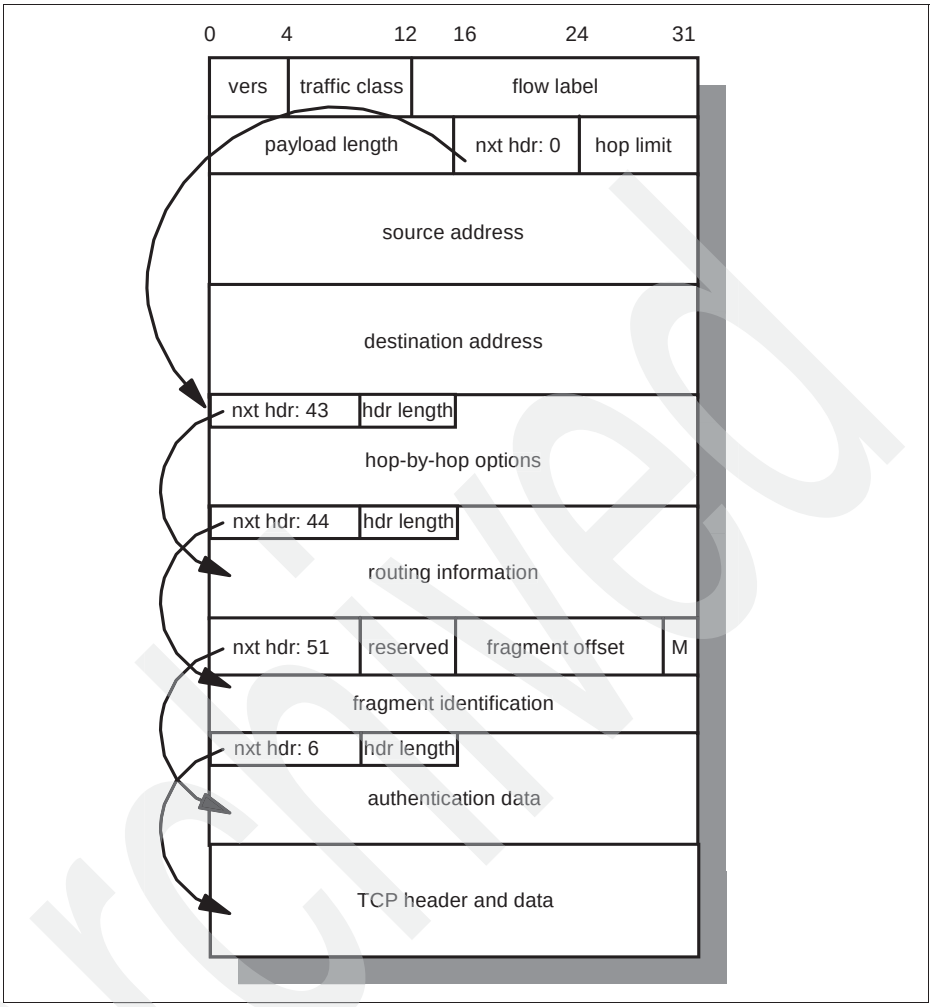


Figure 9-3 IPv6 packet containing multiple extension headers

The length of each header varies, depending on type, but is always a multiple of 8 bytes. There are a limited number of IPv6 extension headers, any one of which can be present only once in the IPv6 packet (with the exception of the Destination Options Header, 60, which can appear more than once). IPv6 nodes that originate packets are required to place extension headers in a specific order (numeric order, with the exception of 60), although IPv6 nodes that receive packets are not required to verify that this is the case. The order is important for efficient processing at intermediate routers. Routers will generally only be interested in the hop-by-hop options and the routing header. After the router has read this far, it does not need to read further in the packet and can immediately

forward. When the Next Header field contains a value other than one for an extension header, this indicates the end of the IPv6 headers and the start of the higher-level protocol data.

IPv6 allows for encapsulation of IPv6 within IPv6 (“tunneling”). This is done with a Next Header value of 41 (IPv6). The encapsulated IPv6 packet can have its own extension headers. Because the size of a packet is calculated by the originating node to match the path MTU, IPv6 routers should not add extension headers to a packet, but instead encapsulate the received packet within an IPv6 packet of their own making (which can be fragmented if necessary).

With the exception of the hop-by-hop header (which must immediately follow the IP header if present), extension headers are not processed by any router on the packet's path except the final one.

Hop-by-hop header

A hop-by-hop header contains options that must be examined by every node the packet traverses, as well as the destination node. It must immediately follow the IPv6 header (if present) and is identified by the special value 0 in the Next Header field of the IPv6 basic header. (This value is not actually a protocol number but a special case to identify this unique type of extension header).

Hop-by-hop headers contain variable length options of the format shown in Figure 9-4 (commonly known as the *Type-Length-Value (TLV)* format).

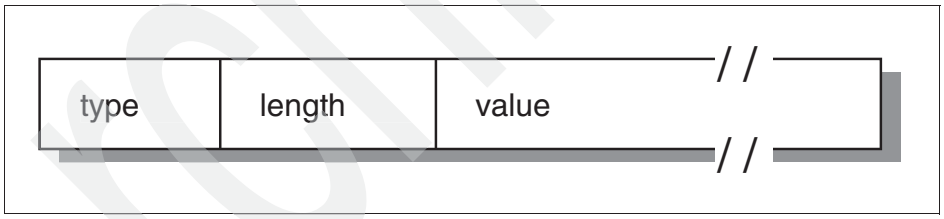


Figure 9-4 IPv6 Type-Length-Value (TLV) option format

Archive

Where:

Type The type of the option. The option types all have a common format (Figure 9-5).

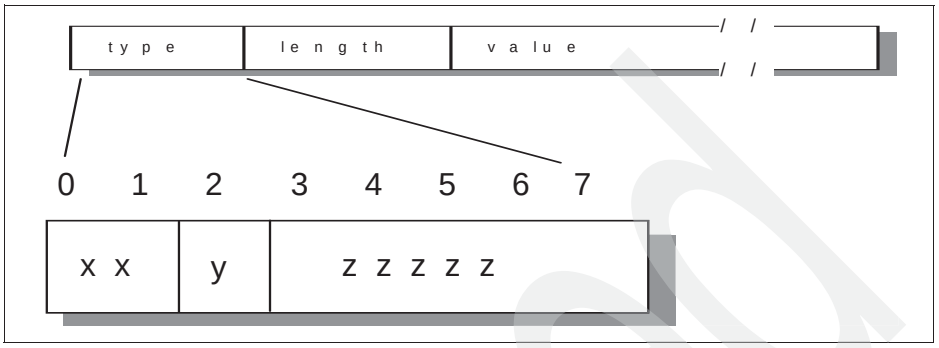


Figure 9-5 IPv6 Type-Length-Value (TLV) option type format

Where:

xx A 2-bit number, indicating how an IPv6 node that does not recognize the option should treat it:

- **0** Skip the option and continue.
- **1** Discard the packet quietly.
- **2** Discard the packet and inform the sender with an ICMP Unrecognized Type message.
- **3** Discard the packet and inform the sender with an ICMP Unrecognized Type message unless the destination address is a multicast address.

y If set, this bit indicates that the value of the option might change en route. If this bit is set, the entire Option Data field is excluded from any integrity calculations performed on the packet.

zzzzz The remaining bits define the option:

- **0** Pad1
- **1** PadN
- **194** Jumbo Payload Length

Length The length of the option value field in bytes.

Value The value of the option. This is dependent on the type.

Archived

Hop-by-hop header option types

You might have noticed that each extension header is an integer multiple of 8 bytes long in order to retain 8-byte alignment for subsequent headers. This is done not purely for “neatness” but because processing is much more efficient if multibyte values are positioned on natural boundaries in memory (and today's processors have natural word sizes of 32 or 64 bits).

In the same way, individual options are also aligned so that multibyte values are positioned on their natural boundaries. In many cases, this will result in the option headers being longer than otherwise necessary, but still allow nodes to process packets more quickly. To allow this alignment, two padding options are used in hop-by-hop headers:

Pad1 A X'00' byte used for padding a single byte. For longer padding sequences, use the PadN option.

PadN An option in the TLV format (described earlier). The length byte gives the number of bytes of padding after the minimum two that are required.

The third option type in a hop-by-hop header is the *Jumbo Payload Length*. This option is used to indicate a packet with a payload size in excess of 65,535 bytes (which is the maximum size that can be specified by the 16-bit Payload Length field in the IPv6 basic header). When this option is used, the Payload Length in the basic header must be set to zero. This option carries the total packet size, less the 40 byte basic header. See Figure 9-6 for details.

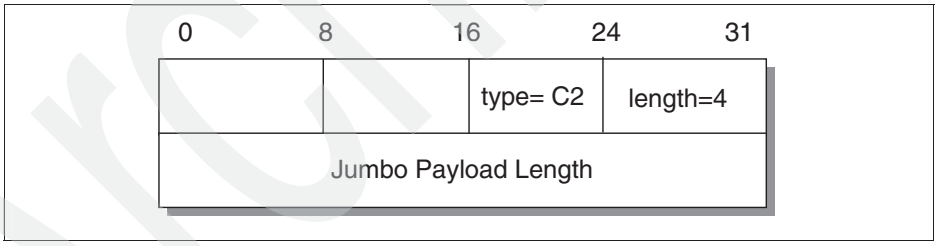


Figure 9-6 Jumbo Payload Length option

Routing header

The path that a packet takes through the network is normally determined by the network itself. Sometimes, however, the source wants more control over the route taken by the packet. It might want, for example, for certain data to take a slower but more secure route than would normally be taken. The routing header (see Figure 9-7 on page 338) allows a path through the network to be predefined. The routing header is identified by the value 43 in the preceding Next Header field. It has its Next Header field as the first byte and a single byte routing type as the second byte. The only type defined initially is type 0, strict/loose

source routing, which operates in a similar way to source routing in IPv4 (see “IP datagram routing options” on page 105).

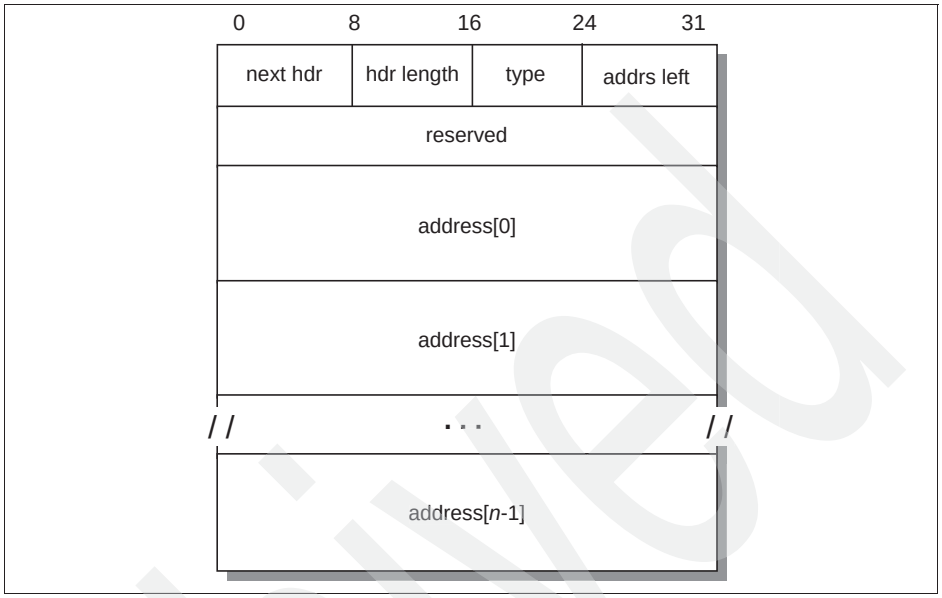


Figure 9-7 IPv6 routing header

Where:

Next hdr The type of header after this one.

Hdr length Length of this routing header, not including the first 8 bytes.

Type Type of routing header. Currently, this can only have the value 0, meaning strict/loose source routing.

Segments left Number of route segments remaining, that is, number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

Address 1..n A series of 16-byte IPv6 addresses that make up the source route.

The first hop on the required path of the packet is indicated by the destination address in the basic header of the packet. When the packet arrives at this address, the router swaps the next address from the router extension header with the destination address in the basic header. The router also decrements the segments left field by one, and then forwards the packet.

Fragment header

The source node determines the maximum transmission unit or MTU for a path before sending a packet. If the packet to be sent is larger than the MTU, the packet is divided into pieces, each of which is a multiple of 8 bytes and carries a fragment header. We provide details about the fragmentation header in “IPv6 packet fragmentation” on page 351.

Authentication header

The authentication header is used to ensure that a received packet has not been altered in transit and that it really came from the claimed sender. The authentication header is identified by the value 51 in the preceding Next Header field. For the format of the authentication header and further details about authentication, refer to 9.2.5, “IPv6 security” on page 347.

Encapsulating Security Payload

The Encapsulated Security Payload (ESP) is a special extension header, in that it can appear anywhere in a packet between the basic header and the upper layer protocol. All data following the ESP header is encrypted. For further details, see 9.2.5, “IPv6 security” on page 347.

Destination options header

This has the same format as the hop-by-hop header, but it is only examined by the destination node or nodes. Normally, the destination options are only intended for the final destination only and the destination options header will be immediately before the upper-layer header. However, destination options can also be intended for intermediate nodes, in which case, they must precede a routing header. A single packet can, therefore, include two destination options headers. Currently, only the Pad1 and PadN types of options are specified for this header (see “Hop-by-hop header” on page 335). The value for the preceding Next Header field is 60.

9.2.2 IPv6 addressing

The IPv6 address model is specified in RFC 4291 – IP Version 6 Addressing Architecture. IPv6 uses a 128-bit address instead of the 32-bit address of IPv4. That theoretically allows for as many as 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses. Even when used with the same efficiency as today's IPv4 address space, that still allows for 50,000 addresses per square meter of land on Earth.

The IPv6 address provides flexibility and scalability:

- It allows multilevel subnetting and allocation from a global backbone to an individual subnet within an organization.

- It improves multicast scalability and efficiency through scope constraints.
- It adds a new address for server node clusters, where one server can respond to a request to a group of nodes.
- The large IPv6 address space is organized into a hierarchical structure to reduce the size of backbone routing tables.

IPv6 addresses are represented in the form of eight hexadecimal numbers divided by colons, for example:

```
FE80:0000:0000:0000:0001:0800:23E:F5DB
```

To shorten the notation of addresses, leading zeroes in any of the groups can be omitted, for example:

```
FE80:0:0:0:1:800:23E7:F5DB
```

Finally, a group of all zeroes, or consecutive groups of all zeroes, can be substituted by a double colon, for example:

```
FE80::1:800:23E7:F5DB
```

Note: The double colon shortcut can be used only once in the notation of an IPv6 address. If there are more groups of all zeroes that are not consecutive, only one can be substituted by the double colon; the others have to be noted as 0.

The IPv6 address space is organized using format prefixes, similar to telephone country and area codes, that logically divide it in the form of a tree so that a route from one network to another can easily be found. The following prefixes have been assigned so far (Table 9-1).

Table 9-1 IPv6: Format prefix allocation

Allocation	Prefix (bin)	Start of address range (hex)	Mask length (bits)	Fraction of address space
Reserved	0000 0000	0:: /8	8	1/256
Reserved for NSAP	0000 001	200:: /7	7	1/128
Reserved for IPX	0000 010	400:: /7	7	1/128
Aggregatable global unicast addresses	001	2000:: /3	3	1/8

Allocation	Prefix (bin)	Start of address range (hex)	Mask length (bits)	Fraction of address space
Link-local unicast	1111 1110 10	FE80:: /10	10	1/1024
Site-local unicast	1111 1110 11	FEC0:: /10	10	1/1024
Multicast	1111 1111	FF00:: /8	8	1/256
Total allocation				15%

In the following sections, we describe the types of addresses that IPv6 defines.

Unicast address

A unicast address is an identifier assigned to a single interface. Packets sent to that address will only be delivered to that interface. Special purpose unicast addresses are defined as follows:

- Loopback address (::1): This address is assigned to a virtual interface over which a host can send packets only to itself. It is equivalent to the IPv4 loopback address 127.0.0.1.
- Unspecified address (::):This address is used as a source address by hosts while performing autoconfiguration. It is equivalent to the IPv4 unspecified address 0.0.0.0.
- IPv4-compatible address (::<IPv4_address>): Addresses of this kind are used when IPv6 traffic needs to be tunneled across existing IPv4 networks. The endpoint of such tunnels can be either hosts (automatic tunneling) or routers (configured tunneling). IPv4-compatible addresses are formed by placing 96 bits of zero in front of a valid 32-bit IPv4 address. For example, the address 1.2.3.4 (hex 01.02.03.04) becomes ::0102:0304.
- IPv4-mapped address (::FFFF:<IPv4_address>): Addresses of this kind are used when an IPv6 host needs to communicate with an IPv4 host. This requires a dual stack host or router for header translations. For example, if an IPv6 node wants to send data to host with an IPv4 address of 1.2.3.4, it uses a destination address of ::FFFF:0102:0304.
- Link-local address: Addresses of this kind can be used only on the physical network that to which a host's interface is attached.
- Site-local address: Addresses of this kind cannot be routed into the Internet. They are the equivalent of IPv4 networks for private use (10.0.0.0, 176.16.0.0-176.31.0.0, 192.168.0.0-192.168.255.0).

Archived

Global unicast address format

IPv6 unicast addresses are aggregatable with prefixes of arbitrary bit-length, similar to IPv4 addresses under Classless Inter-Domain Routing.

The latest global unicast address format, as specified in RFC 3587 – IPv6 Address architecture and RFC 4291 – IPv6 Global Unicast Address Format, is expected to become the predominant format used for IPv6 nodes connected to the Internet.

Note: This note is intended for readers who worked on the previous unicast format. For new readers, you can skip this special note.

The historical IPv6 unicast address used a two-level allocation scheme which has been replaced by a coordinated allocation policy defined by the Regional Internet Registries (RIRs). There are two reasons for this major change:

- Part of the motivation for obsoleting the old TLA/NLA structure is technical; for instance, there is concern that TLA/NLA is not the technically best approach at this stage of the deployment of IPv6.
- Another part of the reason for new allocation of IPv6 addresses is related to policy and to the stewardship of the IP address space and routing table size, which the RIRs have been managing for IPv4.

The Subnet Local Aggregator (SLA) field in the original UNicast Address Structure remains in function, but with a different name called “subnet ID,” which we describe later in the unicast address format.

Figure 9-8 shows the general format for IPv6 global unicast addresses.

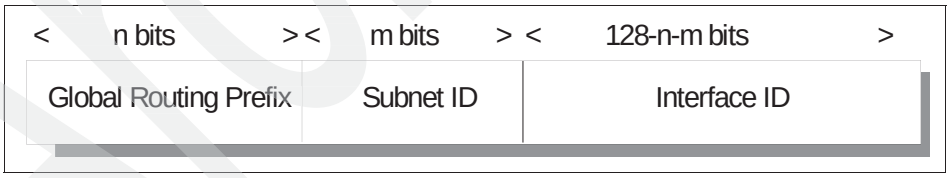


Figure 9-8 Global unicast address format

Where:

Global Routing Prefix

A value assigned to a site for a cluster of subnets/links. The global routing prefix is designed to be structured hierarchically by the RIRs and ISPs.

Subnet ID An identifier of a subnet within the site.The subnet field is designed to be structured hierarchically by site administrators.

Interface ID Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link. They are required to be unique within a subnet prefix. Do not assign the same interface identifier to different nodes on a link. They can also be unique over a broader scope. In some cases, an interface's identifier will be derived directly from that interface's link layer address. The same interface identifier can be used on multiple interfaces on a single node as long as they are attached to different subnets.

All unicast addresses, except those that start with binary value 000, have interface IDs that are 64 bits long and constructed in Modified EUI-64 format.

Multicast address

A multicast address is an identifier assigned to a set of interfaces on multiple hosts. Packets sent to that address will be delivered to all interfaces corresponding to that address. (See 3.3, “Internet Group Management Protocol (IGMP)” on page 119 for more information about IP multicasting.) There are no broadcast addresses in IPv6, their function being superseded by multicast addresses. Figure 9-9 shows the format of an IPv6 multicast address.

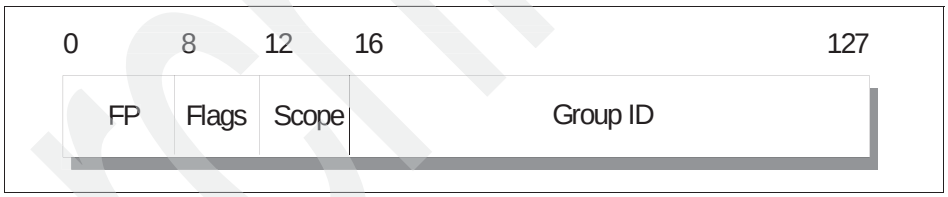


Figure 9-9 IPv6 multicast address format

Where:

FP Format Prefix: 1111 1111.

Flags Set of four flag bits. Only the low order bit currently has any meaning, as follows:

0000 Permanent address assigned by a numbering authority.

0001 Transient address. Addresses of this kind can be established by applications as required. When the application ends, the address will be released by the application and can be reused.

Scope 4-bit value indicating the scope of the multicast. Possible values are:

0	Reserved
1	Confined to interfaces on the local node (node-local)
2	Confined to nodes on the local link (link-local)
5	Confined to the local site
8	Confined to the organization
E	Global scope
F	Reserved

Group ID Identifies the multicast group.

For example, if the NTP servers group is assigned a permanent multicast address, with a group ID of &hex.101, then:

- FF02::101 means all NTP servers on the same link as the sender
- FF05::101 means all NTP servers on the same site as the sender

Certain special purpose multicast addresses are predefined as follows:

FF01::1 All interfaces node-local. Defines all interfaces on the host itself.

FF02::1 All nodes link-local. Defines all systems on the local network.

FF01::2 All routers node-local. Defines all routers local to the host itself.

FF02::2 All routers link-local. Defines all routers on the same link as the host.

FF05::2 All routers site-local. Defines all routers on the same site as the host.

FF02::B Mobile agents link-local.

FF02::1:2 All DHCP agents link-local.

FF05::1:3 All DHCP servers site-local.

For a more complete listing of reserved multicast addresses, see the IANA documentation– IPv6 Multicast Addresses (Assignments.). That document also defines a special multicast address known as the *solicited node address*, which has the format FF02::1:FFxx:xxxx, where xx xxxx is taken from the last 24-bits of a nodes unicast address. For example, the node with the IPv6 address of 4025::01:800:100F:7B5B belongs to the multicast group FF02::1:FF 0F:7B5B. The solicited node address is used by ICMP for neighbor discovery and to detect

duplicate addresses. See 9.3, “Internet Control Message Protocol Version 6 (ICMPv6)” on page 352 for further details.

Anycast address

An anycast address is a special type of unicast address that is assigned to interfaces on multiple hosts. Packets sent to such an address will be delivered to the nearest interface with that address. Routers determine the nearest interface based upon their definition of distance, for example, hops in case of RIP or link state in case of OSPF.

Anycast addresses use the same format as unicast addresses and are indistinguishable from them. However, a node that has been assigned an anycast address must be configured to be aware of this fact.

RFC 4291 currently specifies the following restrictions on anycast addresses:

- An anycast address must not be used as the source address of a packet.
- Any anycast address can only be assigned to a router.

A special anycast address, the *subnet-router address*, is predefined. This address consists of the subnet prefix for a particular subnet followed by trailing zeroes. This address can be used when a node needs to contact a router on a particular subnet and it does not matter which router is reached (for example, when a mobile node needs to communicate with one of the mobile agents on its “home” subnet).

9.2.3 Traffic class

The 8-bit traffic class field allows applications to specify a certain priority for the traffic they generate, thus introducing the concept of *class of service*. This enables the prioritization of packets, as in Differentiated Services. For a comparison of how priority traffic can be handled in an IPv4 network, see 8.1, “Why QoS?” on page 288.

The structure of the traffic class field is illustrated in Figure 9-10 on page 346, where:

DSCP Differentiated Services Code Point (6 bits)

It provides various code sets to mark the per-hop behavior for a packet belonging to a service class.

ECN Explicit Congestion Notification (2 bits)

It allows routers to set congestion indications instead of simply dropping the packets. This avoids delays in retransmissions, while allowing active queuing management.

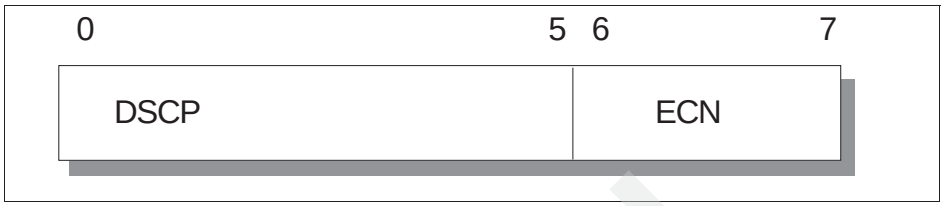


Figure 9-10 Traffic class field

9.2.4 Flow labels

IPv6 introduces the concept of a *flow*, which is a series of related packets from a source to a destination that requires a particular type of handling by the intervening routers, for example, real-time service. The nature of that handling can either be conveyed by options attached to the datagrams (that is, by using the IPv6 hop-by-hop options header) or by a separate protocol (such as Resource Reservation Protocol (RSVP). Refer to “RSVP operation” on page 297.

All packets belonging to the same flow must be sent with the same source address, destination address, and flow label. The handling requirement for a particular flow label is known as the *state information*; this is cached at the router. When packets with a known flow label arrive at the router, the router can efficiently decide how to route and forward the packets without having to examine the rest of the header for each packet.

The maximum lifetime of any flow-handling state established along a flow's path must be specified as part of the description of the state-establishment mechanism, for example, the resource reservation protocol or the flow-setup hop-by-hop option. A source must not reuse a flow label for a new flow within the maximum lifetime of any flow-handling state that might have been established for the prior use of that flow label.

There can be multiple active flows between a source and a destination, as well as traffic that is not associated with any flow. Each flow is distinctly labelled by the 24-bit flow label field in the IPv6 packet. A flow is uniquely identified by the combination of a source address and a non-zero flow label. Packets that do not belong to a flow carry a flow label of zero.

A flow label is assigned to a flow by the flow's source node. New flow labels must be chosen (pseudo-)randomly and uniformly from the range 1 to FFFFF hex. The purpose of the random allocation is to make any set of bits within the Flow Label field suitable for use as a hash key by routers for looking up the state associated with the flow.

See RFC 3697 for further details about the use of the flow label.

9.2.5 IPv6 security

There are two optional headers defined for security purposes:

- Authentication Header (AH)
- Encapsulated Security Payload (ESP)

AH and ESP in IPv6 support authentication, data integrity, and optionally confidentiality. AH conveys the authentication information in an IP package, while ESP carries the encrypted data of the IP package.

Either or both can be implemented alone or combined in order to achieve different levels of user security requirements. Note that they can also be combined with other optional header to provision security features. For example, a routing header can be used to list the intermediate secure nodes for a packet to visit on the way, thus allowing the packet to travel only through secure routers.

IPv6 requires support for IPSec as a mandatory standard. This mandate provides a standards-based solution for network security needs and promotes interoperability.

Authentication header

The authentication header is used to ensure that a received packet has not been altered in transit and that it really came from the claimed sender (Figure 9-11). The authentication header is identified by the value 51 in the preceding Next Header field. The format of the authentication header and further details are specified in REF 4302.

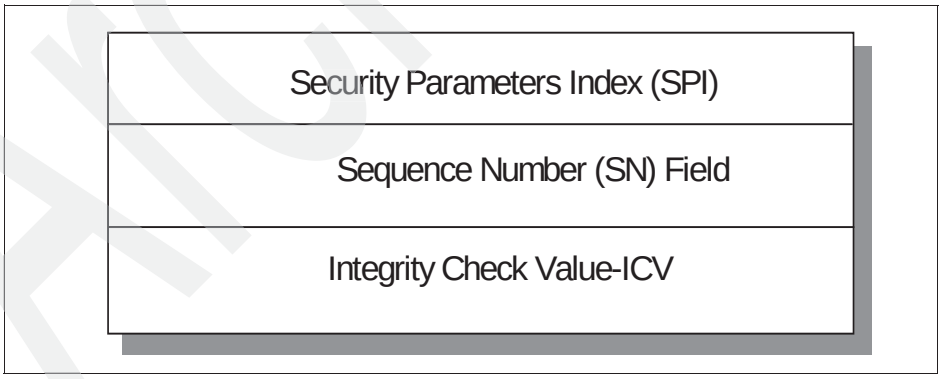


Figure 9-11 IPV6 security authentication header

Where:

Security Parameters Index (SPI)
The SPI is an arbitrary 32-bit value that is used by a

receiver to identify the Security Association (SA) to which an incoming packet is bound.

For a unicast SA, the SPI can be used by itself to specify an SA, or it can be used in conjunction with the IPSec protocol type (in this case AH).The SPI field is mandatory. Traffic to unicast SAs described earlier must be supported by all AH implementations.

If an IPSec implementation supports multicast, it must support multicast SAs using a special de-multiplexing algorithm.

Sequence Number This unsigned 32-bit field contains a counter value that increases by one for each packet sent, that is, a per-SA packet sequence number.

For a unicast SA or a single-sender multicast SA, the sender must increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended.

The field is mandatory and must always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all AH implementations must be capable of performing the processing, Thus, the sender must always transmit this field, but the receiver need not act upon it.

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. The first packet sent using a given SA will have a sequence number of 1; if anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Therefore, the sender's counter and the receiver's counter must be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^{32} packet on an SA.

Extended (64-bit) Sequence Number (ESN)

To support high-speed IPSec implementations, a new option for sequence numbers should be offered, as an extension to the current, 32-bit sequence number field. Use of an Extended Sequence Number (ESN) must be negotiated by an SA management protocol. The ESN feature is applicable to multicast as well as unicast SAs.

Integrity Check Value (ICV)

This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. The field must be an integral multiple of 32 bits (IPv4 or IPv6) in length. All implementations must support such padding and must insert only enough padding to satisfy the IPv4/IPv6 alignment requirements.

Encapsulating Security Payload

The Encapsulated Security Payload (ESP) is defined in RFC 4303. All data following the ESP header is encrypted. Figure 9-12 illustrates the ESP structure with the additional field explained after the figure.

The packet begins with the Security Parameters Index (SPI) and Sequence Number (SN). Following these fields is the Payload Data, which has a substructure that depends on the choice of encryption algorithm and mode and on the use of TFC padding. Following the Payload Data are Padding and Pad Length fields and the Next Header field. The optional Integrity Check Value (ICV) field completes the packet.

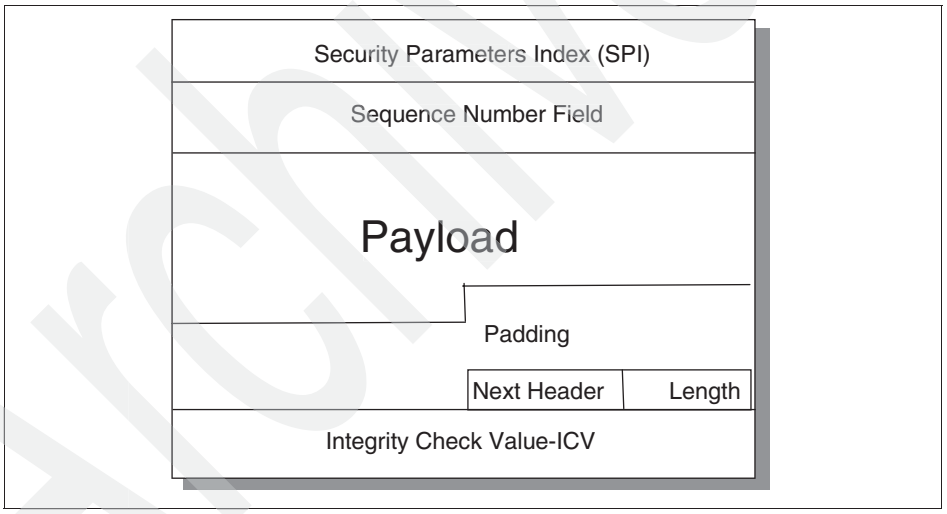


Figure 9-12 IPv6 ESP

Where:

Payload Data Payload Data is a variable-length field containing data (from the original IP packet). It is a mandatory field and is an integral number of bytes in length.

If the algorithm used to encrypt the payload requires cryptographic synchronization data, for example, an Initialization Vector (IV), this data is carried in the Payload field.

Any encryption algorithm that requires an explicit, per-packet synchronization data must indicate the length, any structure for such data, and the location of this data.

If such synchronization data is implicit, the algorithm for deriving the data must be part of the algorithm definition.

Note that the beginning of the next layer protocol header must be aligned relative to the beginning of the ESP header. For IPv6, the alignment is a multiple of 8 bytes.

9.2.6 Packet sizes

All IPv6 nodes are expected to dynamically determine the maximum transmission unit (MTU) supported by all links along a path (as described in RFC 1191 – Path MTU Discovery) and source nodes will only send packets that do not exceed the path MTU. IPv6 routers will, therefore, not have to fragment packets in the middle of multihop routes and allow much more efficient use of paths that traverse diverse physical transmission media. IPv6 requires that every link supports an MTU of 1280 bytes or greater.

Archived

IPv6 packet fragmentation

The source node determines the maximum transmission unit or MTU for a path before sending a packet. If the packet to be sent is larger than the MTU, the packet is divided into pieces, each of which is a multiple of 8 bytes and carries a fragment header. The fragment header is identified by the value 44 in the preceding Next Header field and has the following format (Figure 9-13).

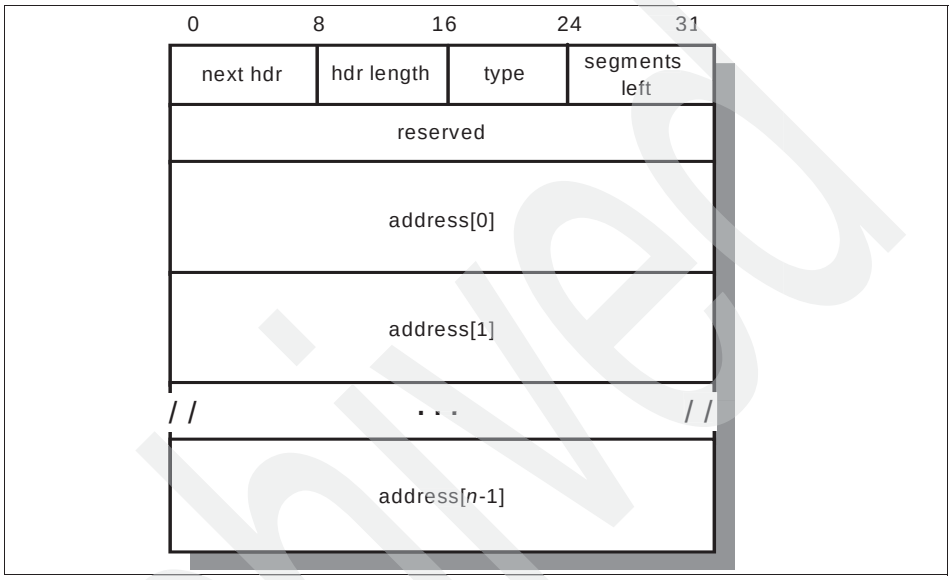


Figure 9-13 IPv6 fragment header

Where:

Nxt hdr The type of next header after this one.

Reserved 8-bit reserved field; initialized to zero for transmission and ignored on reception.

Fragment offset A 13-bit unsigned integer giving the offset, in 8-byte units, of the following data relative to the start of the original data before it was fragmented.

Res 2-bit reserved field; initialized to zero for transmission and ignored on reception.

M More flag. If set, it indicates that this is not the last fragment.

Fragment identification
This is an unambiguous identifier used to identify fragments of the same datagram. This is very similar to the IPv4 Identifier field, but it is twice as wide.

9.3 Internet Control Message Protocol Version 6 (ICMPv6)

IP concerns itself with moving data from one node to another. However, in order for IP to perform this task successfully, there are many other functions that need to be carried out: error reporting, route discovery, and diagnostics, to name a few. All these tasks are carried out by the Internet Control Message Protocol (see 3.2, “Internet Control Message Protocol (ICMP)” on page 109). In addition, ICMPv6 carries out the tasks of conveying multicast group membership information, a function that was previously performed by the IGMP protocol in IPv4 (see 3.3, “Internet Group Management Protocol (IGMP)” on page 119) and address resolution, previously performed by ARP (see 3.4, “Address Resolution Protocol (ARP)” on page 119).

ICMPv6 messages and their use are specified in RFC 4443 – Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification and RFC 2461 – Neighbor Discovery for IP Version 6 (IPv6). Both RFCs are draft standards with a status of elective.

Every ICMPv6 message is preceded by an IPv6 header (and possibly some IP extension headers). The ICMPv6 header is identified by a Next Header value of 58 in the immediately preceding header.

ICMPv6 messages all have a similar format, shown in Figure 9-14.

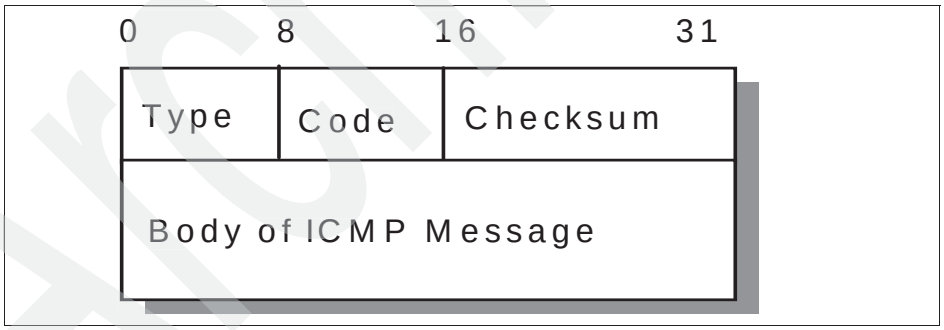


Figure 9-14 ICMPv6 general message format

Where:

Type There are two classes of ICMPv6 messages. Error messages have a Type from 0 to 127. Informational messages have a Type from 128 to 255.

- **1** Destination Unreachable
- **2** Packet Too Big

3	Time (Hop Count) Exceeded
4	Parameter Problem
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect Message

Code Varies according to message type.

Checksum Used to detect data corruption in the ICMPv6 message and parts of the IPv6 header.

Body of message Varies according to message type.

For full details of ICMPv6 messages for all types, refer to RFC 4443.

9.3.1 Neighbor discovery

Neighbor discovery is an ICMPv6 function that enables a node to identify other hosts and routers on its links. The node needs to know of at least one router so that it knows where to forward packets if a target node is not on its local link. Neighbor discovery also allows a router to redirect a node to use a more appropriate router if the node has initially made an incorrect choice.

Archive

Address resolution

Figure 9-15 shows a simple Ethernet LAN segment with four IPv6 workstations.

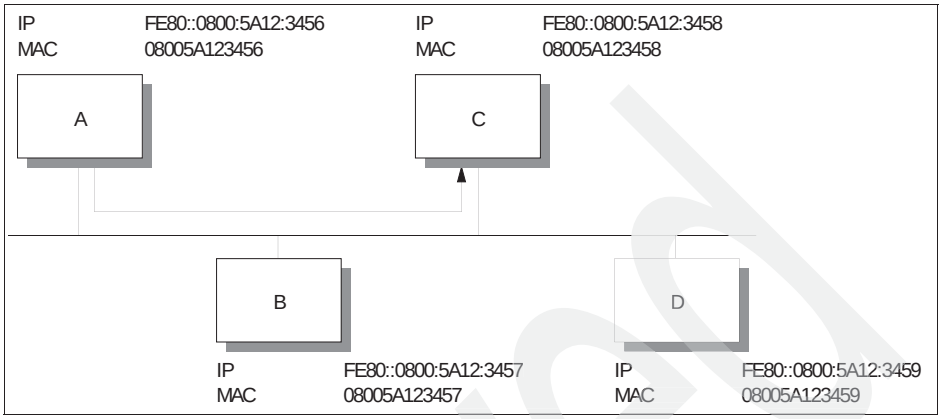


Figure 9-15 IPv6 address resolution example

Workstation A needs to send data to workstation B. It knows the IPv6 address of workstation B, but it does not know how to send a packet, because it does not know its MAC address. To find this information, it sends a *neighbor solicitation* message, of the format shown in Figure 9-16.

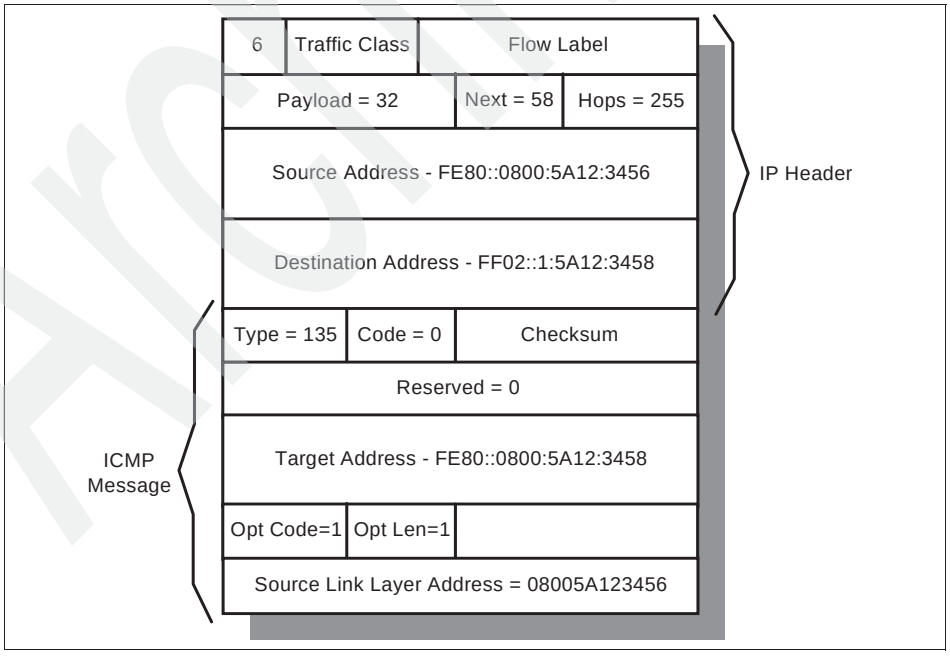


Figure 9-16 Neighbor solicitation message format

Notice the following important fields in the IP header of this packet:

Next 58 (for the following ICMP message header).

Hops Any solicitation packet that does *not* have hops set to 255 is discarded. This ensures that the solicitation has not crossed a router.

Destination address This address is the *solicited node address* for the target workstation (a special type of multicast; see page 344). Every workstation *must* respond to its own solicited node address but other workstations will simply ignore it. This is an improvement over ARP in IPv4, which uses broadcast frames that have to be processed by every node on the link.

In the ICMP message itself, notice:

Type 135 (Neighbor Solicitation).

Target address This is the known IP address of the target workstation.

Source link layer address

This is useful to the target workstation and saves it from having to initiate a neighbor discovery process of its own when it sends a packet back to the source workstation.

Archived

The response to the neighbor solicitation message is a *neighbor advertisement*, which has the following format (Figure 9-17).

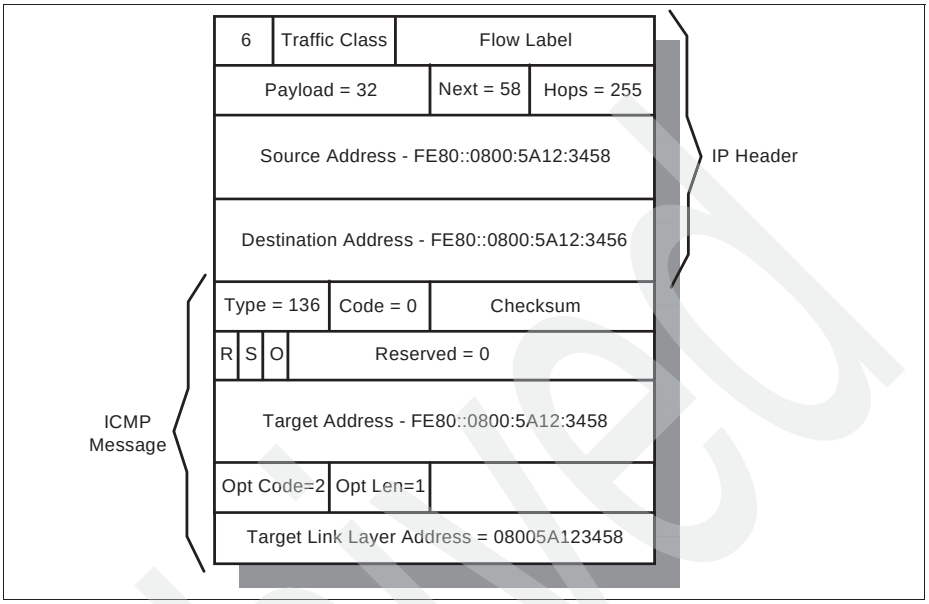


Figure 9-17 Neighbor advertisement message

The neighbor advertisement is addressed directly back to workstation A. The ICMP message option contains the target IP address together with the target's link layer (MAC) address. Note also the following flags in the advertisement message:

R Router flag. This bit is set on if the sender of the advertisement is a router.

S Solicited flag. This bit is set on if the advertisement is in response to a solicitation.

O Override flag. When this bit is set on, the receiving node must update an existing cached link layer entry in its neighbor cache.

After workstation A receives this packet, it commits the information to memory in its neighbor cache, and then forwards the data packet that it originally wanted to send to workstation C.

Neighbor advertisement messages can also be sent by a node to force updates to neighbor caches if it becomes aware that its link layer address has changed.

Archived

Router and prefix discovery

Figure 9-15 on page 354 shows a very simple network example. In a larger network, particularly one connected to the Internet, the neighbor discovery process is used to find nodes on the same link in exactly the same way. However, it is more than likely that a node will need to communicate not just with other nodes on the same link but with nodes on other network segments that might be anywhere in the world. In this case, there are two important pieces of information that a node needs to know:

- The address of a router that the node can use to reach the rest of the world
- The prefix (or prefixes) that define the range of IP addresses on the same link as the node that can be reached without going through a router

Routers use ICMP to convey this information to hosts, by means of *router advertisements*. The format of the router advertisement message is shown in Figure 9-18 on page 358. The message generally has one or more attached options; this example shows all three possible options.

Archived

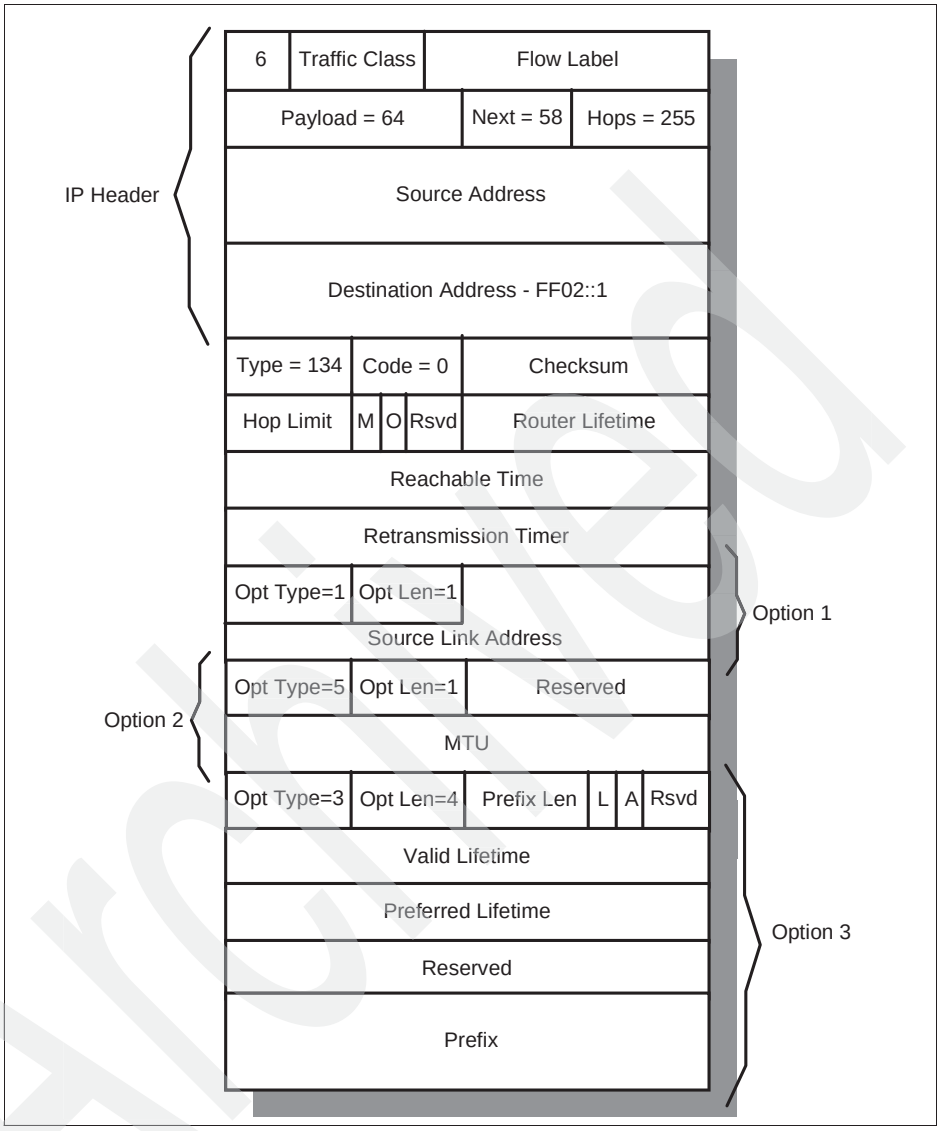


Figure 9-18 Router advertisement message format

Notice the following important fields in the IP header of this packet:

Next 58 (for the following ICMP message header).

Hops Any advertisement packet that does *not* have hops set to 255 is discarded. This ensures that the packet has not crossed a router.

Destination address This address is the special multicast address defining all systems on the local link.

In the ICMP message itself:

Type 134 (router advertisement).

Hop limit The default value that a node should place in the Hop Count field of its outgoing IP packets.

M 1-bit Managed Address Configuration Flag (see “Stateless address autoconfiguration” on page 363).

O 1-bit Other Stateful Configuration Flag (see “Stateless address autoconfiguration” on page 363).

Router lifetime How long the node should consider this router to be available. If this time period is exceeded and the node has not received another router advertisement message, the node should consider this router to be unavailable.

Reachable time This sets a parameter for all nodes on the local link. It is the time in milliseconds that the node should assume a neighbor is still reachable after having received a response to a neighbor solicitation.

Retransmission timer This sets the time, in milliseconds, that nodes should allow between retransmitting neighbor solicitation messages if no initial response is received.

The three possible options in a router advertisement message are:

Option 1 (source link address)
Allows a receiving node to respond directly to the router without having to do a neighbor solicitation.

Option 5 (MTU) Specifies the maximum transmission unit size for the link. For some media, such as Ethernet, this value is fixed, so this option is not necessary.

Option 3 (Prefix) Defines the address prefix for the link. Nodes use this information to determine when they do, and do not, need to use a router. Prefix options used for this purpose have the L (link) bit set on. Prefix options are also used as part of address configuration, in which case the A bit is set on. See “Stateless address autoconfiguration” on page 363 for further details.

A router constantly sends unsolicited advertisements at a frequency defined in the router configuration. A node might, however, want to obtain information about the nearest router without having to wait for the next scheduled advertisement

(for example, a new workstation that has just attached to the network). In this case, the node can send a *router solicitation message*. The format of the router solicitation message is shown in Figure 9-19.

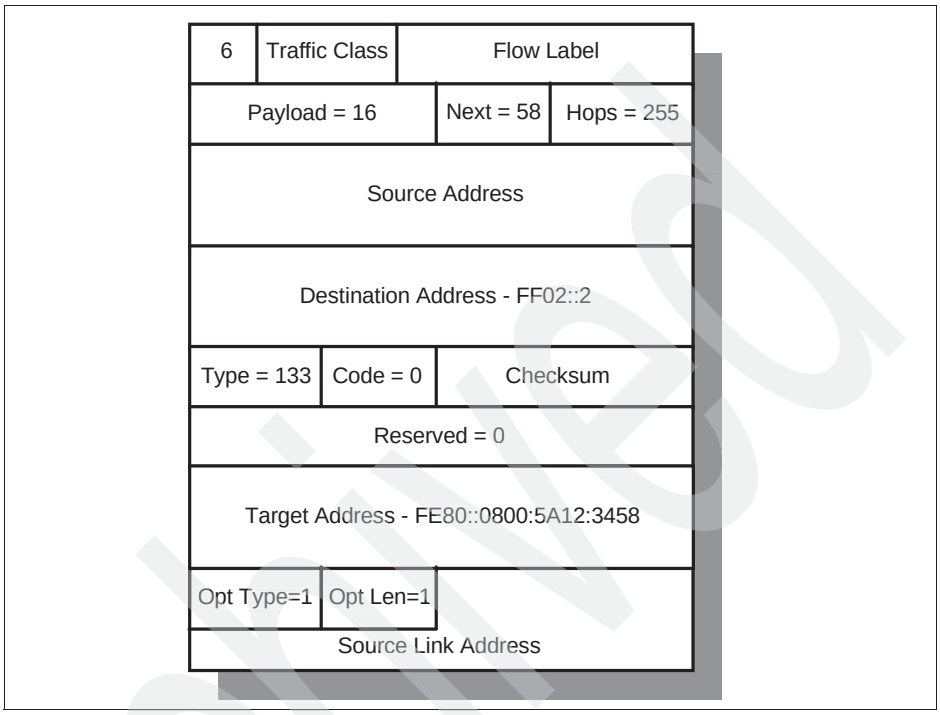


Figure 9-19 Router solicitation message format

Notice the following important fields in the IP header of this packet:

Next 58 (for the following ICMP message header).

Hops Any advertisement packet that does *not* have hops set to 255 is discarded. This ensures that the packet has not crossed a router.

Destination address This address is the special multicast address defining all routers on the local link.

In the ICMP message itself:

Type 133 (Router Solicitation).

Option 1 (source link address)
Allows the receiving router to respond directly to the node without having to do a neighbor solicitation.

Each router that receives the solicitation message responds with a router advertisement sent *directly* to the node that sent the solicitation (not to the all systems link-local multicast address).

Redirection

The router advertisement mechanism ensures that a node will always be aware of one or more routers through which it is able to connect to devices outside of its local links. However, in a situation where a node is aware of more than one router, it is likely that the default router selected when sending data will not always be the most suitable router to select for every packet. In this case, ICMPv6 allows for *redirection* to a more efficient path for a particular destination.

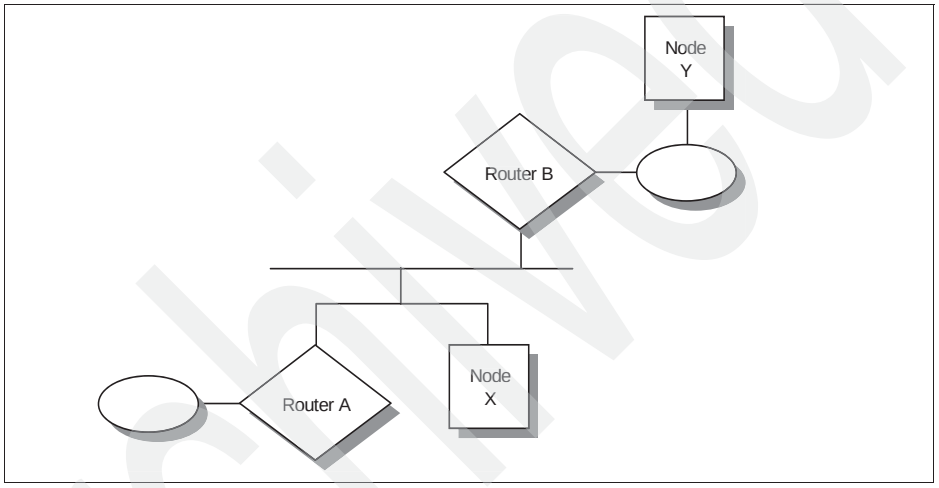


Figure 9-20 Redirection example

Consider the simple example shown in Figure 9-20. Node X is aware of routers A and B, having received router advertisement messages from both. Node X wants to send data to node Y. By comparing node Y's IP address against the local link prefix, node X knows that node Y is not on the local link and that it must therefore use a router. Node X selects router A from its list of default routers and forwards the packet. Obviously, this is not the most efficient path to node Y. As soon as

Archive

router A has forwarded the packet to node Y (through router B), router A sends a *redirect* message to node X. The format of the redirect message (complete with IP header) is shown in Figure 9-21.

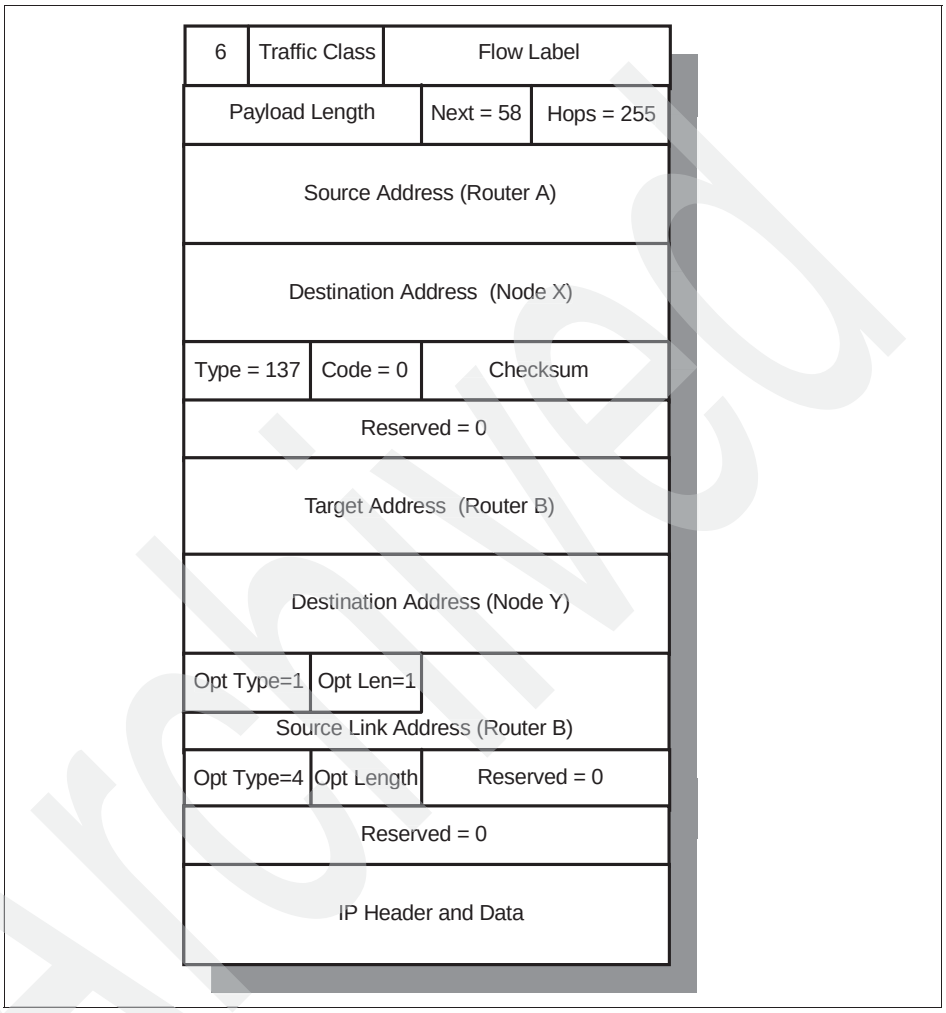


Figure 9-21 Redirect message format

The fields to note in the message are:

Type 137 (Redirect).

Target address This is address of the router that should be used when trying to reach node Y.

Destination address Node Y's IP address.

Option 2 (target link layer address)
Gives link address of router B so that node X can reach it without a neighbor solicitation.

Option 4 (redirected header)
Includes the original packet sent by node X, full IP header, and as much of the data that will fit so that the total size of the redirect message does not exceed 576 bytes.

Neighbor unreachability detection

An additional responsibility of the neighbor discovery function of ICMPv6 is *neighbor unreachability detection* (NUD).

A node actively tracks the reachability state of the neighbors to which it is sending packets. It can do this in two ways: either by monitoring the upper layer protocols to see if a connection is making forward progress (for example, TCP acknowledgments are being received), or issuing specific neighbor solicitations to check that the path to a target host is still available. When a path to a neighbor appears to be failing, appropriate action is taken to try and recover the link. This includes restarting the address resolution process or deleting a neighbor cache entry so that a new router can be tried in order to find a working path to the target.

NUD is used for all paths between nodes, including host-to-host, host-to-router, and router-to-host. NUD can also be used for router-to-router communication if the routing protocol being used does not already include a similar mechanism. For further information about neighbor unreachability detection, refer to RFC 2461.

Stateless address autoconfiguration

Although the 128-bit address field of IPv6 solves a number of problems inherent in IPv4, the size of the address itself represents a potential problem to the TCP/IP administrator. Because of this, IPv6 has been designed with the capability to automatically assign an address to an interface at initialization time, with the intention that a network can become operational with minimal to no action on the part of the TCP/IP administrator. IPv6 nodes generally use autoconfiguration to obtain their IPv6 address. This can be achieved using DHCP (see 9.5, “DHCP in IPv6” on page 371), which is known as *stateful* autoconfiguration, or by *stateless* autoconfiguration, which is a new feature of IPv6 and relies on ICMPv6.

The stateless autoconfiguration process is defined in RFC 2462 – IPv6 Stateless Address Autoconfiguration. It consists of the following steps:

1. During system startup, the node begins the autoconfiguration by obtaining an interface token from the interface hardware, for example, a 48-bit MAC address on token-ring or Ethernet networks.
2. The node creates a tentative link-local unicast address by combining the well-known link-local prefix (FE80::/10) with the interface token.
3. The node attempts to verify that this tentative address is unique by issuing a neighbor solicitation message with the tentative address as the target. If the address is already in use, the node will receive a neighbor advertisement in response, in which case the autoconfiguration process stops. (Manual configuration of the node is then required.)
4. If no response is received, the node assigns the link-level address to its interface. The host then sends one or more router solicitations to the all-routers multicast group. If there are any routers present, they will respond with a router advertisement. If no router advertisement is received, the node attempts to use DHCP to obtain an address and configuration information. If no DHCP server responds, the node continues using the link-level address and can communicate with other nodes on the same link only.
5. If a router advertisement *is* received in response to the router solicitation, this message contains several pieces of information that tells the node how to proceed with the autoconfiguration process (see Figure 9-18 on page 358):
   - M flag: Managed address configuration.

     If this bit is set, the node used DHCP to obtain its IP address.
   - O flag: Other stateful configuration.

     If this bit is set, the node uses DHCP to obtain other configuration parameters.
   - Prefix option: If the router advertisement has a prefix option with the A bit (autonomous address configuration flag) set on, the prefix is used for stateless address autoconfiguration.
6. If stateless address configuration is used, the prefix is taken from the router advertisement and added to the interface token to form the global unicast IP address, which is assigned to the network interface.
7. The working node continues to receive periodic router advertisements. If the information in the advertisement changes, the node must take appropriate action.

Note that it is possible to use both stateless and stateful configuration simultaneously. It is quite likely that stateless configuration will be used to obtain the IP address, but DHCP will then be used to obtain further configuration

information. However, plug-and-play configuration is possible in both small and large networks without the requirement for DHCP servers.

The stateless address configuration process, together with the fact that more than one address can be allocated to the same interface, also allows for the graceful renumbering of all the nodes on a site (for example, if a switch to a new network provider necessitates new addressing) without disruption to the network. For further details, refer to RFC 2462.

9.3.2 Multicast Listener Discovery (MLD)

The process used by a router to discover the members of a particular multicast group is known as *Multicast Listener Discovery* (MLD). MLD is a subset of ICMPv6 and provides the equivalent function of IGMP for IPv4 (see 3.3, “Internet Group Management Protocol (IGMP)” on page 119). This information is then provided by the router to whichever multicast routing protocol is being used so that multicast packets are correctly delivered to all links where there are nodes listening for the appropriate multicast address.

MLD is specified in RFC 2710 – Multicast Listener Discovery (MLD) for IPv6. MLD uses ICMPv6 messages of the format shown in Figure 9-22.

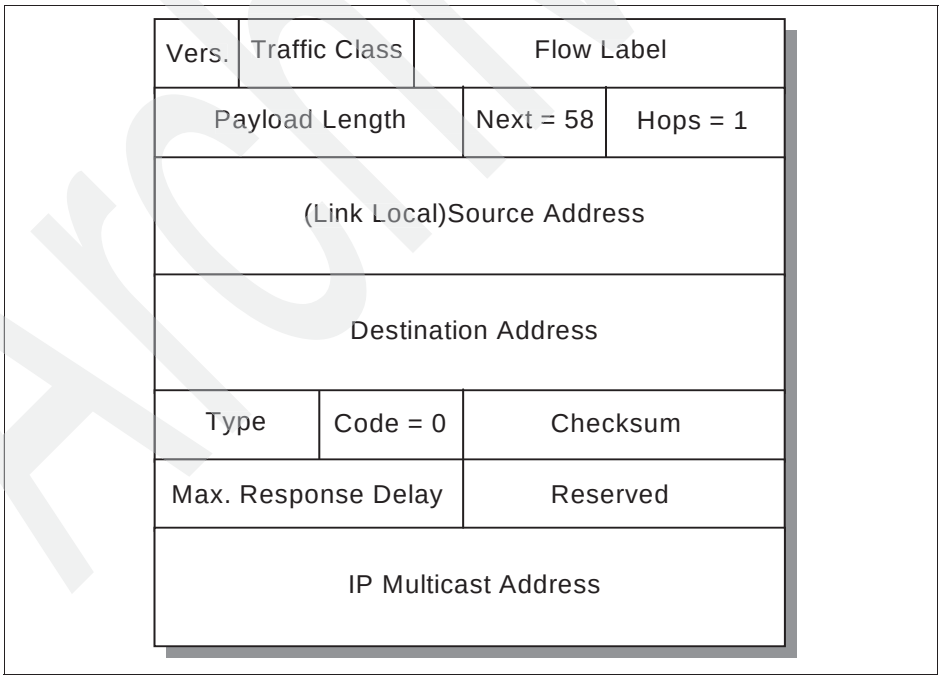


Figure 9-22 MLD message format

Note the following fields in the IPv6 header of the message:

Next 58 (for the following ICMPv6 message header).

Hops Always set to 1.

Source address A link-local source address is used.

In the MLD message itself, notice:

Type There are three types of MLD messages:

130 Multicast Listener Query

There are two types of queries:

- General query: Used to find which multicast addresses are being listened for on a link.
- Multicast-address-specific query: Used to find if any nodes are listening for a specific multicast address on a link.

131 Multicast listener report

Used by a node to report that it is listening to a multicast address.

132 Multicast listener done

Used by a node to report that it is ceasing to listen to a multicast address.

Code Set to 0 by sender and ignored by receivers.

Max response delay This sets the maximum allowed delay before a responding report must be sent. This parameter is only valid in query messages. Increasing this parameter can prevent sudden bursts of high traffic if there a lot of responders on a network.

Multicast address In a query message, this field is set to zero for a general query, or set to the specific IPv6 multicast address for a multicast-address-specific query.

In a response or done message, this field contains the multicast address being listened for.

A router uses MLD to learn which multicast addresses are being listened for on each of its attached links. The router only needs to know that nodes listening for a particular address are present on a link; it does not need to know the unicast address of those listening nodes, or how many listening nodes are present.

A router periodically sends a General Query on each of its links to the all nodes link-local address (FF02::1). When a node listening for any multicast addresses receives this query, it sets a delay timer (which can be anything between 0 and maximum response delay) for each multicast address for which it is listening. As each timer expires, the node sends a *multicast listener report* message containing the appropriate multicast address. If a node receives another node's report for a multicast address while it has a timer still running for that address, it stops its timer and does not send a report for that address. This prevents duplicate reports being sent and, together with the timer mechanism, prevents excess, or bursty traffic being generated.

The router manages a list of, and sets a timer for, each multicast address it is aware of on each of its links. If one of these timers expires without a report being received for that address, the router assumes that no nodes are still listening for that address, and the address is removed from the list. Whenever a report *is* received, the router resets the timer for that particular address.

When a node has finished listening to a multicast address, if it was the last node on a link to send a report to the router (that is, its timer delay was not interrupted by the receipt of another node's report), it sends a *multicast listener done* message to the router. If the node *was* interrupted by another node before its timer expired, it assumes that other nodes are still listening to the multicast address on the link and therefore does not send a done message.

When a router receives a done message, it sends a multicast-address-specific message on the link. If no report is received in response to this message, the router assumes that there are no nodes still listening to this multicast address and removes the address from its list.

9.4 DNS in IPv6

With the introduction of 128-bit addresses, IPv6 makes it even more difficult for the network user to be able to identify another network user by means of the IP address of his or her network device. The use of the Domain Name System (DNS) therefore becomes even more of a necessity.

A number of extensions to DNS are specified to support the storage and retrieval of IPv6 addresses. These are defined in RFC 3596 – DNS Extensions to Support IP Version 6, which is a proposed standard with elective status. However, there is also work in progress on usability enhancements to this RFC, described in an Internet draft of the same name.

The following extensions are specified:

- A new resource record type, AAAA, which maps the domain name to the IPv6 address
- A new domain, which is used to support address-to-domain name lookups
- A change to the definition of existing queries so that they will perform correct processing on both A and AAAA record types

9.4.1 Format of IPv6 resource records

RFC 3596 defines the format of the AAAA record as similar to an A resource record, but with the 128-bit IPv6 address encoded in the data section and a Type value of 28 (decimal).

A special domain, IP6.INT, is defined for inverse (address-to-host name) lookups (similar to the *in-addr.arpa* domain used in IPv4). As in IPv4, the address must be entered in reverse order, but hexadecimal digits are used rather than decimal notation.

For example, for the IPv6 address:

```
2222:0:1:2:3:4:5678:9ABC
```

The inverse domain name entry is:

```
c.b.a.9.8.7.6.5.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.2.2.2.2.IP6.INT.
```

So, if the previous address relates to the node ND1.test.com, we might expect to see the following entries in the name server zone data:

```
$origin test.com.

ND1     99999 IN AAAA 2222:0:1:2:3:4:5678:9ABC

cba98765400030002000100000002222.IP6.INT. IN  PTR ND1
```
[2]

Proposed changes to resource records

The IPv6 addressing system has been designed to allow for multiple addresses on a single interface and to facilitate address renumbering (for example, when a company changes one of its service providers). RFC 3596 – DNS Extensions to Support IP Version 6 proposes changes to the format of the AAAA resource record to simplify network renumbering.

[2] All characters making up the reversed IPv6 address in this PTR entry should be separated by a period(.). These have been omitted in this example for clarity.

Archived

The proposed format of the data section of the AAAA record is shown in Figure 9-23.

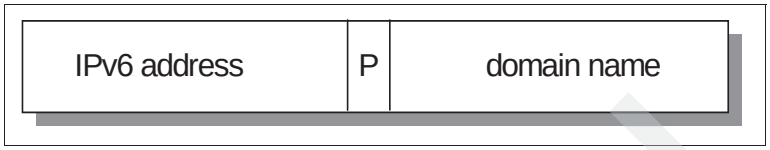


Figure 9-23 AAAA resource record: Proposed data format

Where:

IPv6 address 128-bit address (contains only the lower bits of the address)

P Prefix Length (0-128)

Domain name The domain name of the prefix

To see how this format works, consider the example shown in Figure 9-24.

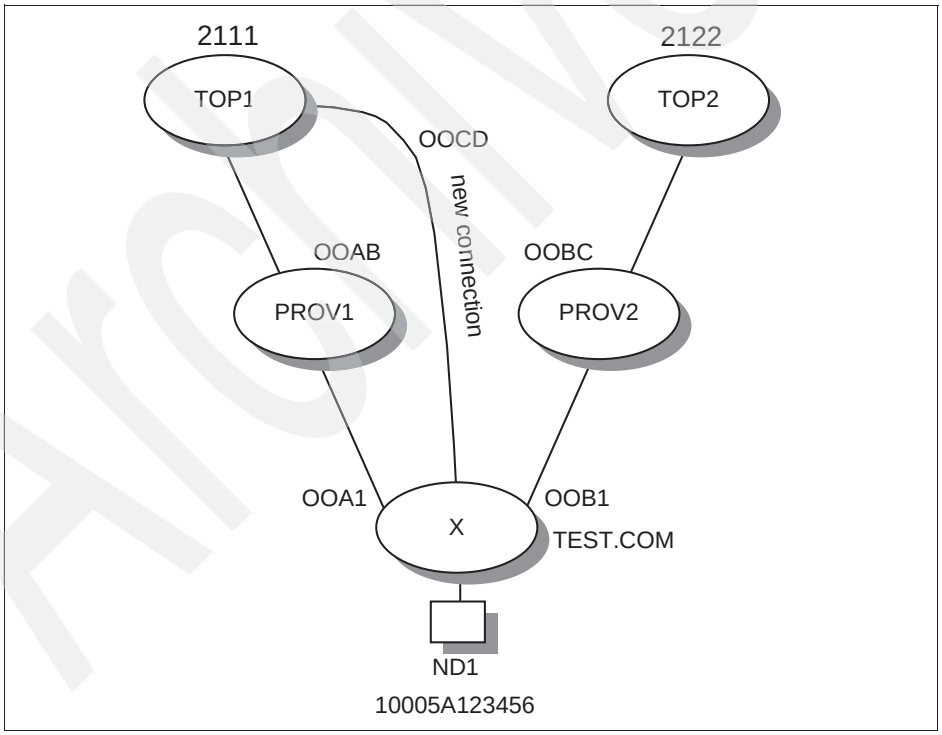


Figure 9-24 Prefix numbering example

Archived

Site X is multihomed to two providers, PROV1 and PROV2. PROV1 gets its transit services from top-level provider TOP1. PROV2 gets its service from TOP2. TOP1 has the top-level aggregate (TLA ID + format prefix) of 2111. TOP2 has the TLA of 2222.

TOP1 has assigned the next-level aggregate (NLA) of 00AB to PROV1. PROV2 has been assigned the NLA of 00BC by TOP2.

PROV1 has assigned the subscriber identifier 00A1 to site X. PROV2 has assigned the subscriber identifier 00B1 to site X.

Node ND1, at site X, which has the interface token of 10005A123456, is therefore configured with the following two IP addresses:

```
2111:00AB:00A1::1000:5A12:3456
2222:00BC:00B1::1000:5A12:3456
```

Site X is represented by the domain name test.com. Each provider has their own domain, top1.com, top2.com, prov1.com, and prov2.com. In each of these domains, an IP6 subdomain is created that is used to hold prefixes. The node ND1 can now be represented by the following entries in the DNS:

```
ND1.TEST.COM AAAA ::1000:5A12:3456 80
IP6.TEST.COM

IP6.TEST.COM AAAA 0:0:00A1:: 32 IP6.PROV1.COM
IP6.TEST.COM AAAA 0:0:00B1:: 32 IP6.PROV2.COM

IP6.PROV1.COM AAAA 0:00AB:: 16 IP6.TOP1.COM

IP6.PROV2.COM AAAA 0:00BC:: 16 IP6.TOP2.COM

IP6.TOP1.COM AAAA 2111::

IP6.TOP2.COM AAAA 2222::
```

This format simplifies the job of the DNS administrator considerably and makes renumbering changes much easier to implement. Say, for example, site X decides to stop using links from providers PROV1 and PROV2 and invests in a connection direct from the top-level service provider TOP1 (who allocates the next-level aggregate 00CD to site X). The only change necessary in the DNS is for the two IP6.TEST.COM entries to be replaced with a single entry, as follows:

```
IP6.TEST.COM AAAA 0:00CD:: 16 IP6.TOP1.COM
```

9.5 DHCP in IPv6

Although IPv6 introduces stateless address autoconfiguration, DHCP retains its importance as the stateful alternative for those sites that want to have more control over their addressing scheme. Used together with stateless autoconfiguration, DHCP provides a means of passing additional configuration options to nodes after they have obtained their addresses. (See 3.7, “Dynamic Host Configuration Protocol (DHCP)” on page 130 for a detailed description of DHCP.)

The RFC 3315 defines DHCP in IPv6, and RFC 3736 defines stateless DHCP for IPv6.

DHCPv6 has some significant differences from DHCPv4, because it takes advantage of some of the inherent enhancements of the IPv6 protocol. Some of the principal differences include:

- As soon as a client boots, it already has a link-local IP address, which it can use to communicate with a DHCP server or a relay agent.
- The client uses multicast addresses to contact the server, rather than broadcasts.
- IPv6 allows the use of multiple IP addresses per interface and DHCPv6 can provide more than one address when requested.
- Some DHCP options are now unnecessary. Default routers, for example, are now obtained by a client using IPv6 neighbor discovery.
- DHCP messages (including address allocations) appear in IPv6 message extensions, rather than in the IP header as in IPv4.
- There is no requirement for BOOTP compatibility.
- There is a new reconfigure message, which is used by the server to send configuration changes to clients (for example, the reduction in an address lifetime). Clients must continue to listen for reconfigure messages after they have received their initial configuration.

9.5.1 DHCPv6 messages

The following DHCPv6 messages are currently defined:

DHCP Solicit This is an IP multicast message. The DHCP client forwards the message to FF02::1:2, the well-known multicast address for all DHCP agents (relays and servers). If received by a relay, the relay forwards the message to FF05::1:3, the well-known multicast address for all DHCP servers.

Archived

DHCP Advertise This is a unicast message sent in response to a DHCP Solicit. A DHCP server will respond directly to the soliciting client if on the same link, or through the relay agent if the DHCP Solicit was forwarded by a relay. The advertise message can contain one or more extensions (DHCP options).

DHCP Request After the client has located the DHCP server, the DHCP request (unicast message) is sent to request an address, configuration parameters, or both. The request must be forwarded by a relay if the server is not on the same link as the client. The request can contain extensions (options specified by the client) that can be a subset of all the options available on the server.

DHCP Reply An IP unicast message sent in response to a DHCP request (can be sent directly to the client or through a relay). Extensions contain the address, parameters, or both committed to the client.

DHCP Release An IP unicast sent by the client to the server, informing the server of resources that are being released.

DHCP Reconfigure An IP unicast or multicast message, sent by the server to one or more clients, to inform them that there is new configuration information available. The client must respond to this message with a DHCP request to request these new changes from the server.

For further details about DHCPv6, refer to RFC3315 and RFC 3736.

9.6 IPv6 mobility support

There are some unique requirements in mobile network applications. For example, while a mobile station or mobile node is always logically identified by its home address, it can physically move around in the IPv6 Internet. For a mobile node to remain reachable while moving, each mobile node has to get a temporary address when it is newly attached to a visiting location network.

While situated away from its home, a mobile node is associated with a care-of address, which provides information about the mobile node's current location. IPv6 packets addressed to a mobile node's home address are transparently routed to its care-of address. IPv6 mobile network cache the binding of a mobile node's home address with its care-of address, and then send any packets destined for the mobile node directly to it at this care-of address.

At any traveling location, there are always multiple service providers for competition in the wireless market, and multiple network prefixes are available. Mobile IPv6 provides binding support for the address to an attached visiting network. The native IPv6 routing header also supports route selection for a packet to go through desired networks. The capability allows network service provider selection. And as a result, it enforces a security and service policy by going through only authorized gateway service nodes.

There are certain enhancements in IPv6 that are particularly well suited to the mobile environment, including:

- A mobile node uses a temporary address while away from home location. It can use the IPv6 Destination Optional header to store its home address. An intended destination can access the field to get the mobile node's home address for substitution when processing the packet.
- A mobile station can list the all routing header for the packets to follow a particular paths in order to connect to a selective service provider network.
- Also, most packets sent to a mobile node while it is away from its home location can be tunneled by using IPv6 routing (extension) headers, rather than a complete encapsulation, as used in Mobile IPv4, which reduces the processing cost of delivering packets to mobile nodes.
- Unlike Mobile IPv4, there is no requirement for routers to act as "foreign agents" on behalf of the mobile node, because neighbor discovery and address autoconfiguration allow the node to operate away from home without any special support from a local router.
- The dynamic home agent address discovery mechanism in Mobile IPv6 returns a single reply to the mobile node. The directed broadcast approach used in IPv4 returns separate replies from each home agent.

To better use the native IPv6 capabilities in next generation (3G) wireless network and service, the IPv6 working group and 3rd Generation Partnership Project (or 3GPP) working group has conducted joint discussions. As a result of adopting native IPv6 features (for example, IPv6 address prefix allocation and so on), they ensure that handsets are compatible with mobile computers in sharing drivers and related software.

Archived

On top of the native IPv6 support to mobility, standard extensions are added to ensure that any nodes, whether mobile or stationary, can communicate efficiently with a mobile node. Additional Mobile IPv6 features include:

- Mobile IPv6 allows a mobile node to move from one link to another without changing the mobile node's “home address.” Packets can be routed to the mobile node using this address regardless of the mobile node's current point of attachment to the Internet. The mobile node can also continue to communicate with other nodes (stationary or mobile) after moving to a new link. The movement of a mobile node away from its home link is thus transparent to transport and higher-layer protocols and applications.
- The Mobile IPv6 protocol is just as suitable for mobility across homogeneous media as for mobility across heterogeneous media. For example, Mobile IPv6 facilitates node movement from one Ethernet segment to another as well as node movement from an Ethernet segment to a wireless LAN cell, with the mobile node's IP address remaining unchanged in spite of such movement.
- You can think of the Mobile IPv6 protocol as solving the network layer mobility management problem. Some mobility management applications, for example, handover among wireless transceivers, each of which covers only a very small geographic area, have been solved using link layer techniques. As another example, in many current wireless LAN products, link layer mobility mechanisms allow a “handover” of a mobile node from one cell to another, re-establishing link layer connectivity to the node in each new location.

  Note: In mobility terminology, a handover deals with moving from a cellular to another cellular. But the concept can be generalized into a wireless-wireline integration environment. For example:

  - Layer-2 handover provides a process by which the mobile node changes from one link layer connection to another in a change of a wireless or wireline access point.
  - Subsequent to an L2 handover, a mobile node detects a change in an on-link subnet prefix that requires a change in the primary care-of address.

- Mobile IPv6 route optimization avoids congestion of the home network by getting a mobile node and a corespondent node to communicate directly. Route optimization can operate securely even without prearranged Security Associations.
- Support for route optimization is a fundamental part of the protocol, rather than as a nonstandard set of extensions. It is expected that route optimization can be deployed on a global scale between all mobile nodes and correspondent nodes.

- The IPv6 Neighbor Unreachability Detection assures symmetric reachability between the mobile node and its default router in the current location. Most packets sent to a mobile node while away from home in Mobile IPv6 are sent using an IPv6 routing header rather than IP encapsulation, increasing efficiencies when compared to Mobile IPv4.
- Mobile IPv6 is decoupled from any particular link layer, because it uses IPv6 Neighbor Discovery instead of Address Resolution Protocol (ARP). This also improves the robustness of the protocol.
- Mobile IPv6 defines a new IPv6 protocol, using the Mobility header to carry the following messages:
  - Home Test Init
  - Home Test
  - Care-of Test Init
  - Care-of Test

    These four messages perform the return routability procedure from the mobile node to a correspondent node.
  - Binding Update and Acknowledgement

    A Binding Update is used by a mobile node to notify a node or the mobile node's home agent of its current binding. The Binding Update sent to the mobile node's home agent to register its primary care-of address is marked as a “home registration.”
  - Binding Refresh Request

    A Binding Refresh Request is used by a correspondent node to request that a mobile node reestablish its binding with the correspondent node. The association of the home address of a mobile node with a “care-of” address for that mobile node remains for the life of that association.
- Mobile IPv6 also introduces four new ICMP message types, two for use in the dynamic home agent address discovery mechanism, and two for renumbering and mobile configuration mechanisms:
  - The following two new ICMP message types are used for home agent address discovery: Home Agent Address Discovery Request and Home Agent Address Discovery Reply.
  - The next two message types are used for network renumbering and address configuration on the mobile node: Mobile Prefix Solicitation and Mobile Prefix Advertisement.

In summary, IPv6 provides native support for mobile applications. Additional extensions have also been added to Mobile IPv6 protocols. IETF has been cooperating with other standard organizations such as 3GPP in Mobile IPv6.

Archived

For further information about mobility support in IPv6, refer to RFC 3775.

9.7 IPv6 new opportunities

IPv6 opens up new opportunities in infrastructure and services as well as in research opportunities.

9.7.1 New infrastructure

As new internet appliances are added into the IP world, the Internet becomes a new infrastructure in multiple dimensions:

- IPv6 can serve as the next generation wireless core network infrastructure. As described in 9.6, “IPv6 mobility support” on page 372, various capabilities in security, addressing, tunneling and so on have enabled mobility applications.
- Additional sensor devices can be connected into the IPv6 backbone with an individual IP address. Those collective sensor networks will become part of the fabric in IPv6 network infrastructure.
- “Smart” networks with sufficient bandwidth and quality of service make the Internet available for phone calls and multimedia applications. We expect that next generation IPv6 network will replace traditional telephone network to become the dominant telecommunication infrastructure.
- As virtualization is widely deployed in both computing data centers and network services, the IPv6 functions become mandatory in security, in flow label processing, and so on. Next generation data centers and network services will evolve around the IPv6 platforms.
- IPv6 can create a new virtual private network (VPN) infrastructure, with inherently built-in tunneling capabilities. It also decouples security boundaries from the organization perimeter in the security policy. We expect that network virtualization is possible with IPv6 VPN on demand provisions and management.
- Inside a computer, the traditional I/O bus architecture might be replaced by a pure IP packet exchanged structure. This scheme might further improve the network computing infrastructure by separating the computing and storage components physically.

Archived

9.7.2 New services

The basic features and new functions in IPv6 provide stimulation to new services creation and deployment. Here are some high-level examples. We encourage you to refer to Part 3, “Advanced concepts and new technologies” on page 721 for more details.

- Presence Service (refer to Chapter 19, “Presence over IP” on page 707) can be developed on top of Location Based Service (LBS). For example, in pure LBS, movie theaters can post attractive title advertisements to a patron’s mobile device when entering the movie zone. In PS, users can setup additional preferences and other policy attributes. As a result, the underlying network services can be aware of user preference and privacy requirements. So, rather than pushing the advertisement to all patrons in the movie zone, those advertisements have to be filtered and tailored accordingly to “do-not-disturb” or “category-specific” preferences.
- Anonymous Request Service (ARS) can be developed by exploiting the new IPv6 address allocation functions. For example, a location address can use a random but unique link ID to send packets in reporting ethical or policy violations within an enterprise or in government services.
- Voice and Video over IP (which we call V^2oIP in IPv6) will replace traditional phone service and provide video services over IPv6. For details about VoIP, refer to Chapter 20, “Voice over Internet Protocol” on page 723. For details about IPTV, refer to Chapter 21, “Internet Protocol Television” on page 745.
- Always On Services (AOS) allows V^2oIPv6 to be ready for service with ease of use. Communication sessions can be kept alive and active using IPv6 mobility functions as well as the IPv6 QoS capability. The “always on” availability is independent of location, movement, or infrastructure.
- On-demand Routing Services (ORS) eliminates routing table updates for unused routes, balancing slow-path and fast-path processing especially in V^2oIPv6 environment.
- IPv6 Management Service (IMS) provides address automatic inventory, service provisioning, and service assurance services.
- IPv6 Operation Service (IOS) supplies on demand configuration, logging, diagnosis, and control services.
- IPv6 Testing Service (ITS) provides capabilities in functional conformance and performance testing for implementations of IETF IPv6 standards or RFCs. Interoperability testing is also a key ITS service.

9.7.3 New research and development platforms

In addition to new opportunities for users and network service vendors, there are IPv6 research opportunities for educational and research and development institutions as well. For example:

- Historically, one of the IETF IP next generation (IPng) project was the development of the 6Bone, which is an Internet-wide virtual network, layered on top of the physical IPv4 Internet. The 6Bone consists of many islands supporting IPv6 packets, linked by tunnels across the existing IPv4 backbone. The 6Bone was widely used for testing of IPv6 protocols and products.

  By June 6th, 2006 the 6Bone was phased out per agreements with the IETF IPv6 community.

  For more information, see:

  `http://www.6Bone.net`

- The 6NET project demonstrated that growth of the Internet can be met using new IPv6 technology. 6NET built a native IPv6-based network connecting 16 European countries. The network allows IPv6 service testing and interoperability with enterprise applications.

  For more information, see:

  `http://www.6net.org`

- Internet2 built an experimental IPv6 infrastructure. The Internet2 consortium (not a network) established IPv6 working group to perform research and education in the following areas:
  - Infrastructure engineering, operations, and deployment
  - Education for campus network engineers
  - Exploring the motivation for use of IPv6

  For more information, see:

  `http://ipv6.internet2.edu`

- Another regional IPv6 example is the MOONv6 project. Moonv6 is just one of the world's largest native IPv6 networks in existence.

  For more information, see:

  `http://www.moonv6.org/`

New open research problems in IPv6 include:

- IPv6 and next generation network architecture design: While IPv6 and associated protocols have solved problems of message specification and control management, the architecture of the next generation IPv6 network itself is still under experiment.

- Network infrastructure and service management: Peer-to-peer (P2P) network applications are available to flood the Internet. However, there is a lack of network and service management and control capability. While we should maintain the access and openness of the Internet, the business and commercial reality in the IP space require fundamental rethinking about network and service management infrastructure support.
- Security: In addition to the native security functions supplied in IPv6 protocols, IPv6 network security architecture needs to define how to extend security across upper layers of IP networks:
  - An integrated security infrastructure combines application security policies to underlying network security capabilities.
  - An integrated security infrastructure also combines content protection into a distribution and transport security layer.
- Real-time control capability: IPv6 quality of service features provide real-time support of voice and multimedia applications. Additional research topics include signaling and integration with IP multimedia subsystems.
- IPv6 network virtualization: Automatic configuration inventory and provisioning capabilities have to be studied in order to allocate networking resources and transport on demand.

9.8 Internet transition: Migrating from IPv4 to IPv6

If the Internet is to realize the benefits of IPv6, a period of transition will be necessary when new IPv6 hosts and routers are deployed alongside existing IPv4 systems. RFC 2893 – Transition Mechanisms for IPv6 Hosts and Routers and RFC2185 – Routing Aspects of IPv6 Transition define a number of mechanisms to be employed to ensure both compatibility between old and new systems and a gradual transition that does not impact the functionality of the Internet. These techniques are sometimes collectively termed *Simple Internet Transition (SIT)*. The transition employs the following techniques:

- Dual-stack IP implementations for hosts and routers that must interoperate between IPv4 and IPv6.
- Imbedding of IPv4 addresses in IPv6 addresses. IPv6 hosts will be assigned addresses that are interoperable with IPv4, and IPv4 host addresses will be mapped to IPv6.
- IPv6-over-IPv4 tunneling mechanisms for carrying IPv6 packets across IPv4 router networks.

- IPv4/IPv6 header translation.This technique is intended for use when implementation of IPv6 is well advanced and only a few IPv4-only systems remain.

9.8.1 Dual IP stack implementation: The IPv6/IPv4 node

The simplest way to ensure that a new IPv6 node maintains compatibility with existing IPv4 systems is to provide a dual IP stack implementation. An IPv6/IPv4 node can send and receive either IPv6 packets or IPv4 datagrams, depending on the type of system with which it is communicating. The node will have both a 128-bit IPv6 address and a 32-bit IPv4 address, which do not necessarily need to be related. Figure 9-25 shows a dual stack IPv6/IPv4 system communicating with both IPv6 and IPv4 systems on the same link.

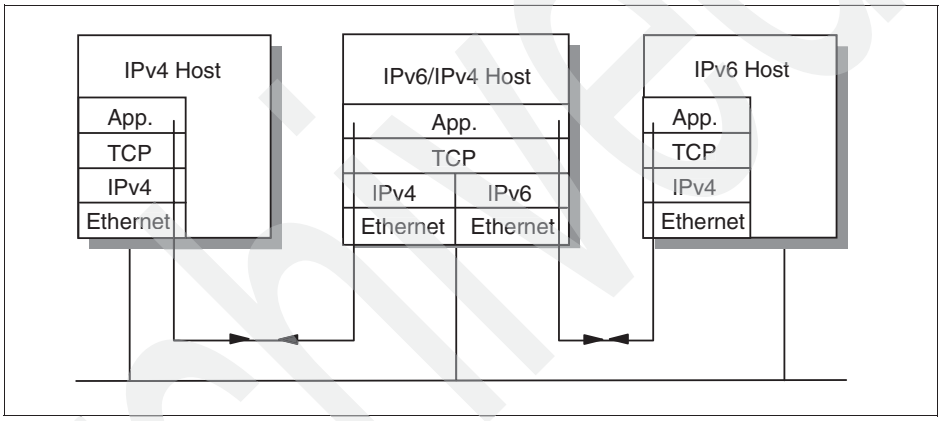


Figure 9-25 IPv6/IPv4 dual stack system

The IPv6/IPv4 node can use stateless or stateful autoconfiguration to obtain its IPv6 address. It can also use any method to obtain its IPv4 address, such as DHCP, BOOTP, or manual configuration. However, if the node is to perform automatic tunneling, the IPv6 address must be an IPv4-compatible address, with the low order 32-bits of the address serving as the IPv4 address. (See 9.2.2, “IPv6 addressing” on page 339.)

Conceptually, the dual stack model envisages a doubling-up of the protocols in the internetwork layer only. However, related changes are obviously needed in all transport-layer protocols in order to operate when using either stack. Application changes are also needed if the application is to exploit IPv6 capabilities, such as the increased address space of IPv6.

When an IPv6/IPv4 node wants to communicate with another system, it needs to know the capabilities of that system and which type of packet it should send. The

DNS plays a key role here. As described in Table 12-2 on page 438, a new resource record type, AAAA, is defined for mapping host names to IPv6 addresses. The results of a name server lookup determine how a node will attempt to communicate with that system. The records found in the DNS for a node depend on which protocols it is running:

- IPv4-only nodes only have A records containing IPv4 addresses in the DNS.
- IPv6/IPv4 nodes that can interoperate with IPv4-only nodes have AAAA records containing IPv4-compatible IPv6 addresses and A records containing the equivalent IPv4 addresses.
- IPv6-only nodes that cannot interoperate with IPv4-only nodes have only AAAA records containing IPv6 addresses.

Because IPv6/IPv4 nodes make decisions about which protocols to use based on the information returned by the DNS, the incorporation of AAAA records in the DNS is a prerequisite to interoperability between IPv6 and IPv4 systems. Note that name servers do not necessarily need to use an IPv6-capable protocol stack, but they must support the additional record type.

9.8.2 Tunneling

When IPv6 or IPv6/IPv4 systems are separated from other similar systems with which they want to communicate by older IPv4 networks, IPv6 packets must be tunneled through the IPv4 network.

IPv6 packets are tunnelled over IPv4 very simply: The IPv6 packet is encapsulated in an IPv4 datagram, or in other words, a complete IPv4 header is added to the IPv6 packet. The presence of the IPv6 packet within the IPv4 datagram is indicated by a protocol value of 41 in the IPv4 header.

There are two kinds of tunneling of IPv6 packets over IPv4 networks: *automatic* and *configured*.

Automatic tunneling

Automatic tunneling relies on IPv4-compatible addresses. The decision of when to tunnel is made by an IPv6/IPv4 host that has a packet to send across an IPv4-routed network area, and it follows the following rules:

- If the destination is an IPv4 or an IPv4-mapped address, send the packet using IPv4 because the recipient is not IPv6-capable. Otherwise, if the destination is on the same subnet, send it using IPv6, because the recipient is IPv6-capable.

- If the destination is not on the same subnet but there is at least one default router on the subnet that is IPv6-capable, or there is a route configured to an IPv6 router for that destination, send it to that router using IPv6. Otherwise, if the address is an IPv4-compatible address, send the packet using automatic IPv6-over-IPv4 tunneling. Otherwise, the destination is a node with an IPv6-only address that is connected through an IPv4-routed area, which is not also IPv6-routed. Therefore, the destination is unreachable.

Note: The IP address must be IPv4-compatible for tunneling to be used. Automatic tunneling cannot be used to reach IPv6-only addresses, because they cannot be addressed using IPv4. Packets from IPv6/IPv4 nodes to IPv4-mapped addresses are not tunnelled to because they refer to IPv4-only nodes.

These rules emphasize the use of an IPv6 router in preference to a tunnel for three reasons:

- There is less inefficiency, because there is no encapsulating IPv4 header.
- IPv6-only features are available.
- The IPv6 routing topology will be used when it is deployed in preference to the pre-existing IPv4 topology.

A node does not need to know whether it is attached to an IPv6-routed or an IPv4-routed area; it will always use an IPv6 router if one is configured on its subnet and will use tunneling if one is not (in which case it can infer that it is attached to an IPv4-routed area).

Automatic tunneling can be either host-to-host, or it can be router-to-host. A source host will send an IPv6 packet to an IPv6 router if possible, but that router might not be able to do the same, and will have to perform automatic tunneling to the destination host itself. Because of the preference for the use of IPv6 routers rather than tunneling, the tunnel will always be as “short” as possible. However, the tunnel will always extend all of the way to the destination host. Because IPv6 uses the same hop-by-hop routing paradigm, a host cannot determine if the packet will eventually emerge into an IPv6-complete area before it reaches the destination host. In order to use a tunnel that does not extend all of the way to the recipient, configured tunneling must be used.

The mechanism used for automatic tunneling is very simple:

1. The encapsulating IPv4 datagram uses the low-order 32 bits of the IPv6 source and destination addresses to create the equivalent IPv4 addresses and sets the protocol number to 41 (IPv6).

2. The receiving node's network interface layer identifies the incoming packets (or packets if the IPv4 datagram was fragmented) as belonging to IPv4 and passes them upward to the IPv4 part of the dual IPv6/IPv4 internetwork layer.
3. The IPv4 layer then receives the datagram in the normal way, reassembling fragments if necessary, notes the protocol number of 41, removes the IPv4 header, and passes the original IPv6 packet “sideways” to the IPv6 part of the internetwork layer.
4. The IPv6 code then processes the original packet as normal. Because the destination IPv6 address in the packet is the IPv6 address of the node (an IPv4-compatible address matching the IPv4 address used in the encapsulating IPv4 datagram), the packet is at its final destination. IPv6 then processes any extension headers as normal and then passes the packet's remaining payload to the next protocol listed in the last IPv6 header.

Figure 9-26 on page 384 shows two IPv6/IPv4 nodes separated by an IPv4 network. Both workstations have IPv4-compatible IPv6 addresses. Workstation A sends a packet to workstation B, as follows:

1. Workstation A has received router solicitation messages from an IPv6-capable router (X) on its local link. It forwards the packet to this router.
2. Router X adds an IPv4 header to the packet, using the IPv4 source and destination addresses derived from the IPv4-compatible addresses. The packet is then forwarded across the IPv4 network, all the way to workstation B. This is *router-to-host* automatic tunneling.
3. The IPv4 datagram is received by the IPv4 stack of workstation B. Because the Protocol field shows that the next header is 41 (IPv6), the IPv4 header is stripped from the datagram and the remaining IPv6 packet is then handled by the IPv6 stack.

Archive

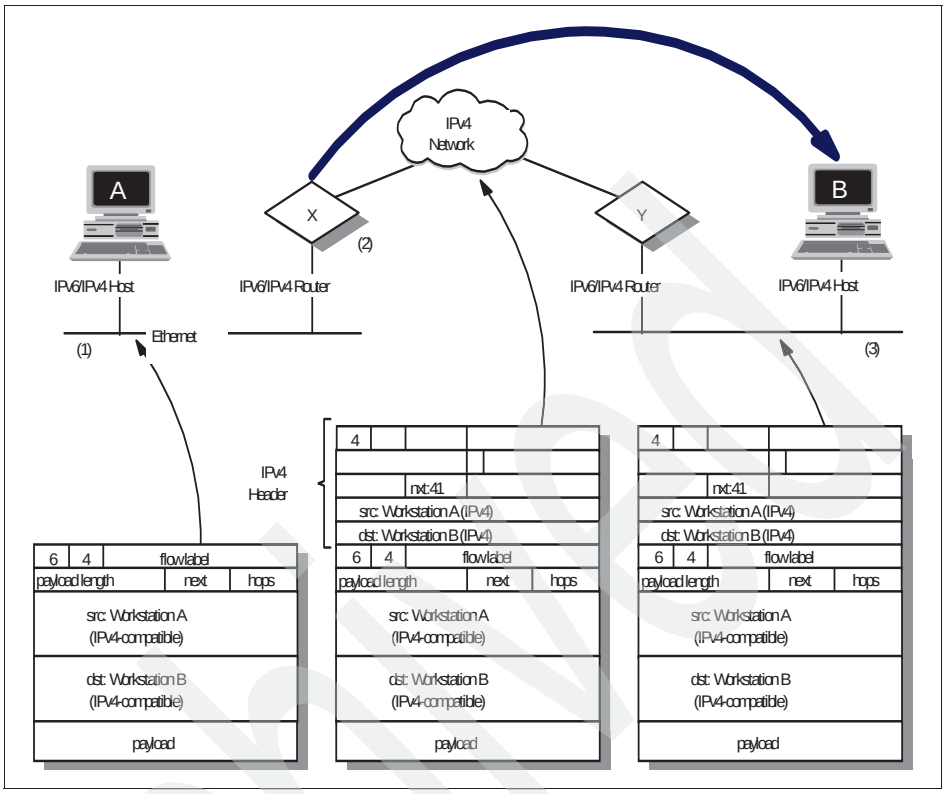


Figure 9-26 Router-to-host automatic tunneling

Figure 9-27 on page 385 shows the host-to-host tunneling scenario. Here workstation B responds as follows:

1. Workstation B has no IPv6-capable router on its local link. It therefore adds an IPv4 header to its own IPv6 frame and forwards the resulting IPv4 datagram directly to the IPv4 address of workstation A through the IPv4 network. This is *host-to-host* automatic tunneling.
2. The IPv4 datagram is received by the IPv4 stack of workstation A. Because the Protocol field shows that the next header is 41 (IPv6), the IPv4 header is stripped from the datagram and the remaining IPv6 packet is then handled by the IPv6 stack.

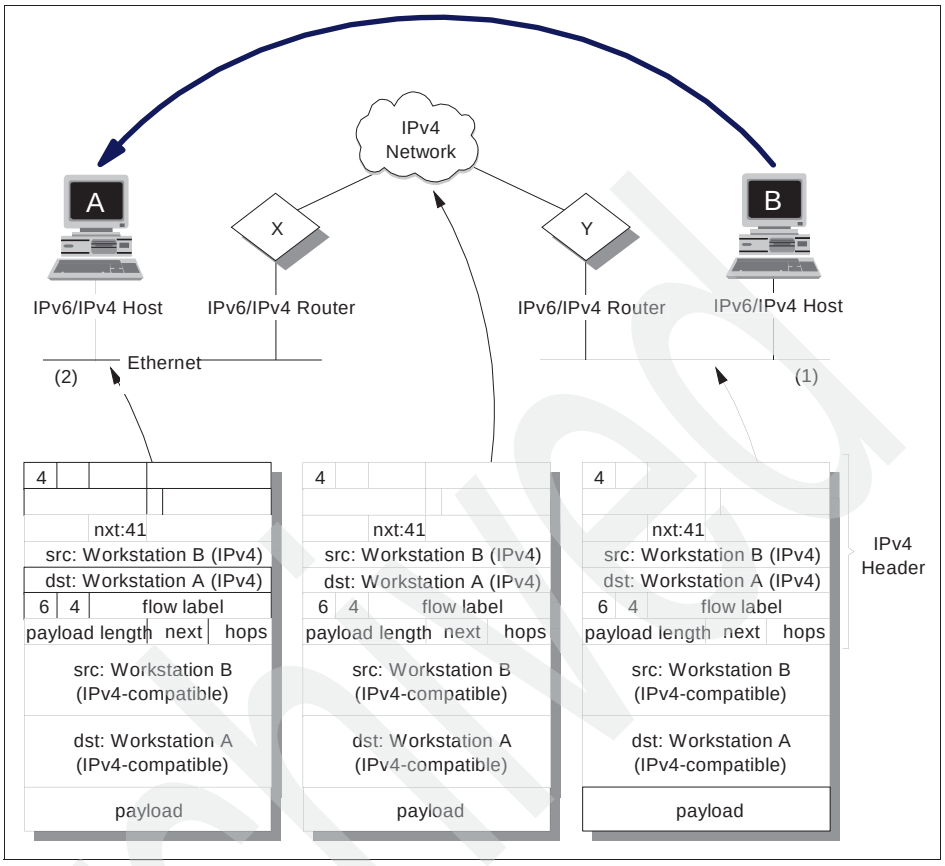


Figure 9-27 Host-to-host automatic tunneling

Configured tunneling

Configured tunneling is used for host-router or router-router tunneling of IPv6-over-IPv4. The sending host or the forwarding router is configured so that the route, as well as having a next hop, also has a *tunnel end* address (which is always an IPv4-compatible address). The process of encapsulation is the same as for automatic tunneling, except that the IPv4 destination address is not derived from the low-order 32 bits of the IPv6 destination address, but from the low-order 32 bits of the tunnel end. The IPv6 destination and source addresses do *not* need to be IPv4-compatible addresses in this case.

When the router at the end of the tunnel receives the IPv4 datagram, it processes it in exactly the same way as a node at the end of an automatic tunnel. When the original IPv6 packet is passed to the IPv6 layer in the router, it recognizes that it is not the destination, and the router forwards the packet on to the final destination as it would for any other IPv6 packet.

It is, of course, possible that after emerging from the tunnel, the IPv6 packet is tunnelled again by another router.

Figure 9-28 on page 387 shows two IPv6-only nodes separated by an IPv4 network. A router-to-router tunnel is configured between the two IPv6/IPv4 routers X and Y.

1. Workstation A constructs an IPv6 packet to send to workstation B. It forwards the packet to the IPv6 router advertising on its local link (X).
2. Router X receives the packet, but has no direct IPv6 connection to the destination subnet. However, a tunnel has been configured for this subnet. The router therefore adds an IPv4 header to the packet, with a destination address of the tunnel-end (router Y) and forwards the datagram over the IPv4 network.
3. The IPv4 stack of router Y receives the frame. Seeing the Protocol field value of 41, it removes the IPv4 header, and passes the remaining IPv6 packet to its IPv6 stack. The IPv6 stack reads the destination IPv6 address, and forwards the packet.
4. Workstation B receives the IP6 packet.

Archive

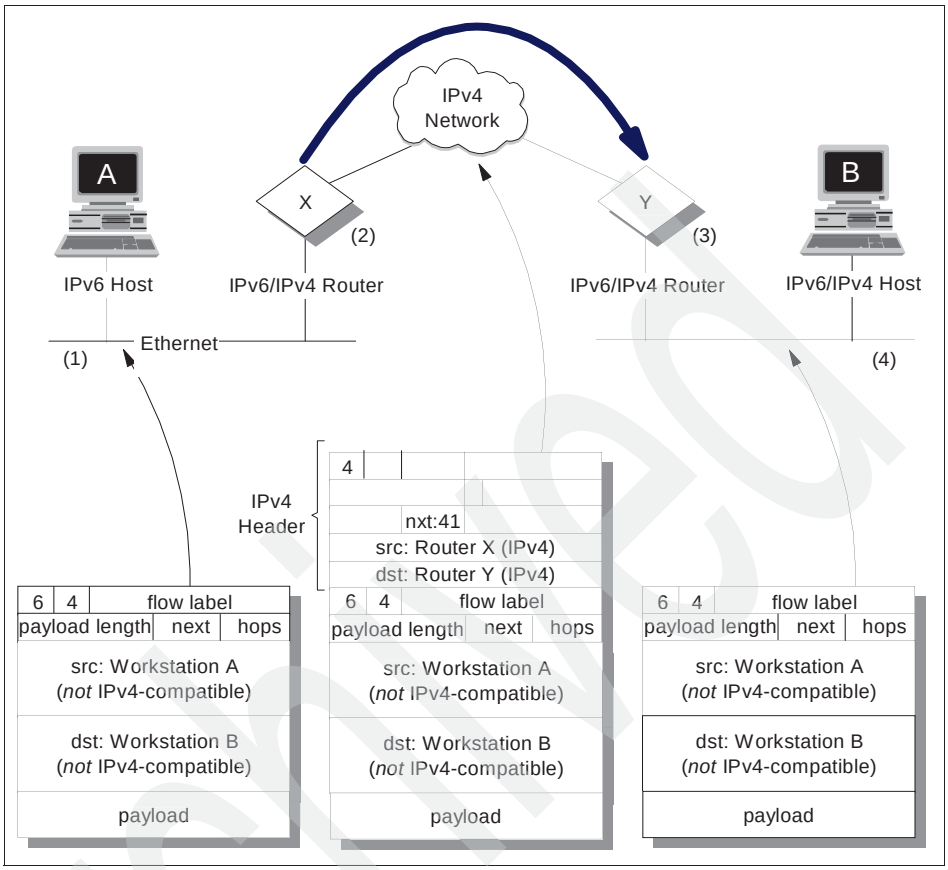


Figure 9-28 Router-to-router configured tunnel

Header translation

Installing IPv6/IPv4 nodes allows for backward compatibility with existing IPv4 systems. However, when migration of networks to IPv6 reaches an advanced stage, it is likely that new systems being installed will be IPv6 only. Therefore, there will be a requirement for IPv6-only systems to communicate with the remaining IPv4-only systems. Header translation is required for IPv6-only nodes to interoperate with IPv4-only nodes. Header translation is performed by IPv6/IPv4 routers on the boundaries between IPv6 routed areas and IPv4 routed areas.

The translating router strips the header completely from IPv6 packets and replaces it with an equivalent IPv4 header (or the reverse). In addition to correctly mapping between the fields in the two headers, the router must convert source and destination addresses from IPv4-mapped addresses to real IPv4 addresses (by taking the low-order 32 bits of the IP address). In the reverse

direction, the router adds the ::FFFF /96 prefix to the IPv4 address to form the IPv4-mapped address. If either the source or the destination IPv6 address is IPv6-only, the header cannot be translated.

Note that for a site with even just one IPv4 host, every IPv6 node with which it needs to communicate must have an IPv4-mapped address.

9.8.3 Interoperability summary

Whether two nodes can interoperate depends on their capabilities and their addresses.

An IPv4 node can communicate with:

- Any IPv4 node on the local link
- Any IPv4 node through an IPv4 router
- Any IPv6 node with IPv4-mapped address through a header translator

An IPv6 node (IPv6-only address) can communicate with:

- Any IPv6 node on the local link
- Any IPv6 node through an IPv6 router on the local link (might require tunneling through the IPv4 network from the router)

An IPv6 node (IPv4-mapped address) can communicate with:

- Any IPv6 node on the local link
- Any IPv6 node through an IPv6 router on the local link (might require tunneling through the IPv4 network from the router)
- Any IPv4 node through a header translator

An IPv6/IPv4 node (IPv4-compatible address) can communicate with:

- Any IPv4 node on the local link
- Any IPv4 node through an IPv4 router on the local link
- Any IPv6 node on the local link
- Any IPv6 node through an IPv6 router on the local link (might require tunneling through the IPv4 network from the router)
- Any IPv6/IPv4 node (IPv4-compatible address) through a host-to-host tunnel

Archived

9.9 RFCs relevant to this chapter

The following RFCs contain detailed information about IPv6:

- RFC 3041 – Privacy Extensions for Stateless Address Autoconfiguration in IPv6 (January 2001)
- RFC 3056 – Connection of IPv6 Domains via IPv4 Clouds (February 2001)
- RFC 3307 – Allocation Guidelines for IPv6 Multicast Addresses (August 2002)
- RFC 3315 – Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (July 2003)
- RFC 3484 – Default Address Selection for Internet Protocol version 6 (IPv6) (February 2003)
- RFC 3596 – DNS Extensions to Support IP Version 6 (October 2003) (Obsoletes RFC3152, RFC1886)
- RFC 3633 – IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6 (December 2003)
- RFC 3646 – DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (December 2003)
- RFC 3697 – IPv6 Flow Label Specification (March 2004)
- RFC 3736 – Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6 (April 2004)
- RFC 3775 – Mobility Support in IPv6 (June 2004)
- RFC 3776 – Using IPSec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents (June 2004)
- RFC 3956 – Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address (November 2004)
- RFC 4007 – IPv6 Scoped Address Architecture (March 2005)
- RFC 4038 – Application Aspects of IPv6 Transition (March 2005)
- RFC 4057 – IPv6 Enterprise Network Scenarios (June 2005)
- RFC 4241 – A Model of IPv6/IPv4 Dual Stack Internet Access Service (December 2005)
- RFC 4443 – Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification (March 2006)
- RFC 4302 – IP Authentication Header (December 2005)
- RFC 4303 – IP Encapsulating Security Payload (ESP) (for v6 and v4) (December 2005)

Archived

- RFC 2675 – IPv6 Jumbograms, August 1999)
- RFC 2460 – Internet Protocol, Version 6 (IPv6) (December 1998)
- RFC 4291 – IP Version 6 Addressing Architecture (February 2006)
- RFC 3587 – IPv6 Global Unicast Address Format (August 2003)
- RFC 2461 – Neighbor Discovery for IP Version 6 (IPv6) (December 1998)
- RFC 2462 – IPv6 Stateless Address Autoconfiguration (December 1998)
- RFC 3596 – DNS Extensions to Support IP Version6 (October 2003)
- RFC 2893 – Transition Mechanisms for IPv6 Hosts and Routers (August 2000)

For more information about any of these topics, see:

- IANA Assignment Documentation: INTERNET PROTOCOL VERSION 6 MULTICAST ADDRESSES, June 2006

  `http://www.iana.org/assignments/ipv6-multicast-addresses`
- Global IPv6 Summit 2006

  `http://www.ipv6.net.cn/2006`
- 6NET

  `http://www.6net.org/`
- IPv6 Working Group

  `http://ipv6.internet2.edu`

Archived

10

Wireless IP

In an increasingly mobile society, the need for wireless connectivity is a consistently growing area. As a result, technology is rapidly advancing to provide wireless support for business and personal use. This chapter discusses some of the fundamental concepts behind wireless IP and the technology that supports it.

© Copyright IBM Corp. 1989-2006. All rights reserved.

10.1 Wireless concepts

Given the diverse nature of wireless implementation, there are a number of terms and concepts relating to the wireless ideology. This section reviews some of the more common of these.

Radio propagation

Radio propagation refers to the behavior exhibited by radio waves as they are transmitted to and from points around the earth, and includes aspects such as aurora, backscatter, and tropospheric scatter.

The decibel (dB)

Signal strength of radio waves is measured in decibels (dBs), specifically by quantifying the amount of signal lost between two points in a wireless network. This measurement is calculated as the difference between a signal's strength at an originating point and at a destination point. Changes in signal strengths are measured in terms of positive or negative dB gain.

Path loss

Path loss refers to a signal's loss in electromagnetic radiation as it propagates from one point to another. Though this reduction can be directly affected by things such as terrain and the environment, the actual loss is inversely proportional to the distance travelled by the signal, and directly proportional to the wave length of the signal.

Effective isotropic radiated power

Effective isotropic radiated power (ERP) is used to quantify the signal strength produced by an antenna. It accounts for both the gain of the antenna as well as the power that feeds into the antenna.

For example, if an antenna has -13 dB gain, and is fed by 100 dB, its ERP is 87 dB, as illustrated in Figure 10-1.

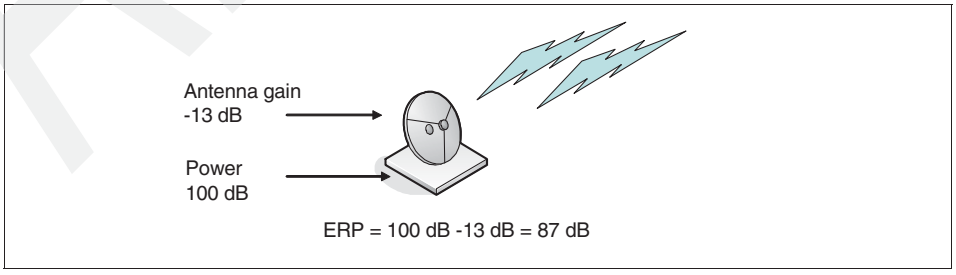


Figure 10-1 ERP example

Fixed versus mobile wireless

There are two types of wireless devices: fixed and mobile. Fixed devices are stationary and draw their power from a utility main. An example of such a device is a wireless router plugged into a wall outlet. Conversely, mobile devices are those that have the capability of movement. Naturally, these are powered from batteries. An example of this is a mobile computer.

Effects of multipath

Similar to a wired IP network, it is possible for radio signals to traverse different paths between a source and destination. This can occur when one signal encounters an obstruction. This can introduce delays into the traversal of signals and is called multipath distortion.

System operating margin

The system operating margin defines the range in which error free reception is achieved. This is calculated in dB as the difference between the received signal level and the receiver's sensitivity. For example, if the received signal is -15 dB, and the sensitivity of the receiver is -10 dB, the system operating margin is 5 dB.

Free space loss

Free space loss is similar to path loss, except that path loss is experienced between any two radio points and thus incorporates signal loss through various types of media. Conversely, free space loss is specific to the lessening of a signal as it traverses free space.

Decibel over isotropic (dBi)

Before decibel isotropic (dBi) units can be understood, the concept of an isotropic antenna must first be explained. An isotropic antenna is theoretical, and produces uniform signal strength in every direction, called isotropic radiation. This sphere can then be used as a point of reference when measuring an actual antenna's strength. This measurement is made in units of dBi, and compares the antenna's strength relative to the isotropic radiation that would be created by an isotropic antenna of the same strength. This is illustrated in Figure 10-2 on page 394.

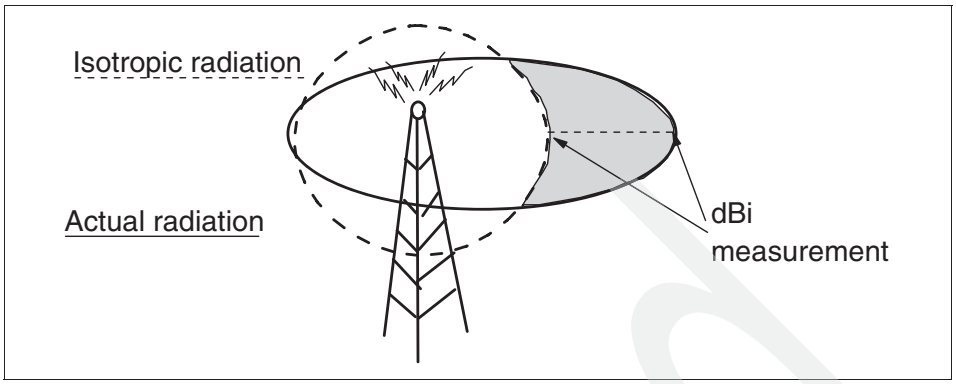


Figure 10-2 Decibel over isotropic

Fresnel zone clearance

When obstructions exist within the path of a signal, diffraction of the signal creates a series of concentric elliptical zones, each zone varying in signal strength. Each of these zones represents a different Fresnel zone within the signal. Fresnel zones are numbered outward from the center, and referred to as the *n*th zone. This is illustrated in Figure 10-3. Note that the first zone has no obstructions, providing the strongest signal to the house. The second zone was created by tree obstructions and carries a signal weaker than the first zone, but stronger than the third. The third zone, with the weakest signal, was the result of an obstructing building.

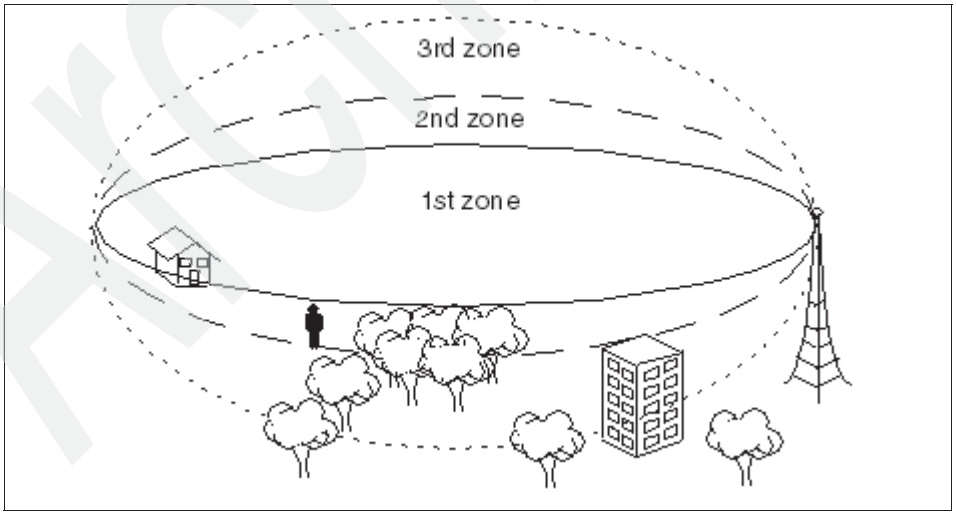


Figure 10-3 An example of Fresnel zones

Line of sight (LOS) and non-line of sight (NLOS) service

Line of sight (LOS) and non-line of sight (NLOS) are used to define a link by its position relative to a signal's transmitter. An LOS link is one that must have an unobstructed path between it and the signal's source, literally meaning that the link has a line of site to the source. This usually indicates that the link is within the first Freznel zone. If a link that requires LOS service moves into the second or third zone (for example, where the person in Figure 10-3 on page 394 is standing), it would no longer have LOS, and might not operate. However, a link that can use NLOS would still operate correctly.

Wireless access point

Wireless access points typically relay data between wireless devices and a wired network. However, multiple access points can be chained together, creating a larger network to allow roaming of mobile devices.

Wireless router

A wireless router acts as a wireless access point combined with an Ethernet hub, forwarding packets between a wireless subnet and any other subnet.

Wireless Ethernet bridge

Wireless Ethernet bridges connect two separate wireless networks without requiring the services of a router.

10.2 Why wireless?

Though the immediate benefit implementing a wireless network (mobility) might seem obvious, there are other benefits that might not be as readily evident.

10.2.1 Deployment and cost effectiveness

When creating a traditional, wired network, much of the construction centers around laying cable. Though this is not as difficult a task when the network is built in parallel with a structure, installing wired networks into existing structures can be quite difficult because the wires must often be installed behind or above solid walls or ceilings. This can incur substantial costs, both in purchasing the wire as well as in paying for the construction to install the wire. When installed, there is also the cost of maintaining the wires, which can degrade over time.

Conversely, creating a wireless network requires minimum construction, if any at all. When building a large-scale network, there might be some initial cost and construction to build antennas, access points, and so on. However, once built,

Archived

the maintenance required by such structures is minimal. Additionally, there is no cost for laying cable, which is significant on a large-scale network.

For small-scale networks (such as office buildings), the cost is relatively minimal. Only access points (such as wireless routers) need to be purchased, and can create their own network or be hooked into an existing network. There is no construction cost, no cost for wiring, and therefore no cost in installing the wiring. Additionally, such a network can be set up and configured in as fast as a day, depending on the complexity of the organization's needs.

10.2.2 Reachability

Wired networks do not lend themselves to certain geographies. For example, imagine laying cable to provide connectivity between research stations in the Amazon, or to interconnect remote communities in sparsely populated regions of Wyoming. Not only would the wiring be costly, but the terrain through which the cable must be laid might be prohibitive. For example, wet or hot climates (such as the Amazon) might cause cabling to deteriorate too fast. Rocky terrains might not be cost effective to bury the cable. Additionally, when the distance between connected points is too great, the signal might degrade before the distance is spanned. This, of course, can be resolved using repeaters, but this adds additional costs.

Implementation of a wireless network can overcome these challenges simply because it nullifies the need for wiring. Distances between nodes can be spanned easily and the nuances of a terrain can be overcome. Additionally, if a wired network is desired, wireless can be used to interconnect remote wired networks.

10.2.3 Scalability

A common challenge faced by growing businesses is outgrowing their network. When first constructing a network, a young business might not have an accurate forecast of the network size needed to accommodate the organization. Then, as the business needs grow, the network is no longer capable of supporting its needs. As described previously, adding additional wiring might be cost prohibitive and might compromise the success of the business.

In such a scenario, wireless networks can offer two solutions. First, wireless capability can be added to an existing wired network. This allows the network to grow as needed, and additions can continue to be made if the needs continue to grow. Second, if the business initially builds a wireless network, the problematic scenario will never occur because the organization can continue to add wireless capability to address growing needs.

10.2.4 Security

One concern over any network is the question of security. As data becomes more sensitive, and more readily available online, the need to protect this data increases rapidly. A common misconception is that hackers or malicious users are facilitated by the growing use of wireless because this allows them to steal data having only proximity to a network.

However, with such a concern in mind, the wireless architectures and technologies were designed specifically with security in mind. As such, wireless networks are often more secure, through the use of advanced authentication and encryption methods, than their wired counterparts.

10.2.5 Connectivity and reliability

Depending on the design and configuration of a wireless network, it is possible that such a network might be prone to the same connectivity outages as a wired network. However, this is a limitation of the design of a particular network and not of the wireless architecture itself. For example, wireless networking lends itself to the concept of mesh networking, described in 10.5.3, “Mesh networking” on page 402. Through such an implementation, as nodes become available or are removed from a network, the overall wireless network can “heal” itself, and still provide connectivity to all of the other nodes.

10.3 WiFi

The term WiFi is short for Wireless Fidelity and is meant to be used generically when referring to any type of 802.11 network, whether 802.11b, 802.11a, dual-band, and so on. The term originated from the Wi-Fi Alliance.

The 802.11 standard refers to a family of specifications developed by the IEEE for wireless LAN technology. The 802.11 standard specifies an over-the-air interface between a wireless client and a base station or between two wireless clients. The IEEE accepted the specification in 1997.

802.11 family of standards

There are several specifications in the 802.11 family of standards:

802.11 Applies to wireless LANs and provides 1 or 2 Mbps transmission in the 2.4 GHz band using either frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS).

802.11a An extension to 802.11 that applies to wireless LANs and provides up to 54 Mbps in the 5 GHz band. 802.11a uses an orthogonal frequency division multiplexing (OFDM) encoding scheme rather than FHSS or DSSS.

802.11b Also known as 802.11 High Rate or WiFi. An extension to 802.11 that applies to wireless LANs and provides 11 Mbps transmission with fallbacks to 5.5, 2, and 1 Mbps in the 2.4 GHz band. 802.11b uses only DSS. 802.11b was a 1999 ratification to the original 802.11 standard, allowing wireless functionality comparable to Ethernet.

802.11g Applies to wireless LANs and provides 20+ Mbps in the 2.4 GHz band.

For additional information about the 802.11 family of standards, see:

`http://www.ieee802.org/11/`

Operation

WiFi operates as a non-switched Ethernet network. Every 100 ms, Wireless Application Protocols (WAPs) broadcast service set identifiers (SSIDs) using *beacon* packets. Clients who receive these beacons can opt to wirelessly connect to the WAP. This determination is usually established by some combination of the following factors:

- Whether or not the client has been configured to connect to the broadcasted SSID.
- The signal strength of the WAP. In particular, a client might receive two beacons from two different WAPs, each one broadcasting the same SSID. In this instance, the client should opt to connect to the WAP demonstrating the stronger signal.
- The level of encryption offered by a WAP.

Each beacon is broadcast at 1 Mbps, ensuring that any client who receives the beacon at a minimum supports communication at this speed. All of the area to which a WAP beacon can be received is referred to as a *hotspot*. Though WiFi hotspots can be several miles long, such an implementation requires multiple WAPs to overlap their individual hotspots using the same SSID.

WiFi can also be used in peer-to-peer mode, allowing mobile devices to communicate with one another in the absence of a wireless network. Although this method of operation does not provide any sort of connectivity to the Internet, it does lend itself to other applications such as backing up data or gaming.

Security

The airborne nature of WiFi inherently makes it susceptible to security risks. No longer hindered by the need to gain access to a wire, malicious users attempting to capture data transfers must only gain proximity to the intended victim. As such, several encryption protocols have been coupled with WiFi in order to secure the data transferred using WiFi.

Wireless Equivalent Privacy (WEP)

Initially, WEP was used to secure WiFi communications. It uses RC4, or *ARCFOUR*, stream cipher to provide confidentiality. Additionally, WEP employs a 33-bit cyclic redundancy check (CRC-32) to ensure data integrity. However, WEP uses a shared encryption key to which all users must have access in order to authenticate with the WAP. This compromises the security of the network because current hacking technology can decode the key using freely distributed programs. Additionally, WEP security, because it employs a stream cipher, is susceptible to stream cipher attacks. Due to these and other shortcomings, WEP has been outdated by WiFi Protected Access (WPA and WPA2).

WiFi Protected Access (WPA)

Created by the Wi-Fi Alliance, WPA also employs a pass phrase concept similar to that of the WEP implementation. However, WPA uses distributed private keys administered by an 802.1X authentication server.

Note: A public-shared key (PSK) mode can be used, but it is less secure.

Data encryption is again provided through the RC4 stream cipher, which uses a 128-bit key and a 48-bit initialization vector. Security is increased by inserting dynamic key changes using the Temporal Key Integrity Protocol (TKIP). Data integrity is guaranteed using the Message Integrity Code (MIC) algorithm, also called Michael's algorithm.

While this increased security implementation compensates for the faults found previously with WEP, cryptoanalysts have still found weaknesses in the WPA architecture. Specifically, Michael's algorithm was chosen because it still allowed mobile devices using WPA to communicate with access points still using WEP, and vice versa. However, the algorithm is still susceptible to packet forgery attacks. To combat this, WPA was enhanced and expanded into WPA2.

WiFi Protected Access (WPA2)

In WPA2, Michael's algorithm is replaced by the Counter Mode with Cipher Block Chaining Message Authentication Protocol (CCMP). Because CCMP provides both data integrity and key management using the Advanced Encryption Standard (AES, also known as Rijndael), it combines both the data integrity and

confidentiality functions of WPA into one protocol. CCMP is considered fully secure.

10.4 WiMax

Also known as WirelessMAN, the Worldwide Interoperability for Microwave Access (WiMAX) is a digital communications system defined by the IEEE standard 802.16 (most recently approved in 2004). Much like the Wi-Fi Wireless Alliance, WiMAX is monitored by the WiMAX forum, which strives to ensure product compliance with the 802.16 standard and device interoperability.

Similar to the client/server model (see 11.1.1, “The client/server model” on page 408), WiMAX uses the notion of *subscriber* stations and *base* stations. Base stations provide the wireless access and provide the same functions as the WAPs. Subscriber stations are the clients using the wireless access provided by the base station.

802.16 family of standards

There are several specifications in the 802.16 family of standards:

802.16 This applies to enabling last mile wireless broadband access and can be used as an alternative to DSL and cable. This specification is also known as WirelessMAN.

802.16a This specification addresses issues of radio spectrum use. It specifies added support for the 2 to 11 GHz range that provides support for low latency applications such as video and voice. It enables the provision of broadband connectivity without the requirement of direct line of sight (LOS) between the subscriber terminals and the base station (BTS).

802.16b This extends 802.16 by increasing the spectrum to 5 and 6 GHz. This provides quality of service (QoS) for voice and video services.

802.16c This extends 802.16 by representing the 10 to 66 GHz range. This extension also addresses issues such as interoperability, performance evaluation, testing, system profiling, and performance evaluation.

802.16e Also known as Mobile WiMaX. This extends and improves the modulation schemes described in the original/fixed WiMax standard. This allows for fixed wireless and mobile NLOS applications by improving upon the Orthogonal Frequency Division Multiple Access (OFDMA). This should not be confused with 802.20.

For additional information about the 802.16 family of standards, see:

http://www.ieee802.org/16/

Security over WiMax

Similar to WiFi, WiMAX uses WAP2, CCMP, and AES. Additionally, WiMAX provides end-to-end authentication through the Public Key Methodology - Extensible Authentication Protocol (PKM-EAP). This relies on Transport Layer Security (TLS) to provide authentication and confidentiality.

Advantages of WiMAX over WiFi

Like WiFi, WiMAX provides wireless access to mobile devices. However, WiMAX has advantages over WiFi in specific applications. WiFi access points are usually unable to guarantee any quality of service (QoS, see Chapter 8, “Quality of service” on page 287), and as such, QoS-dependent applications, such as VoIP (see 20.1, “Voice over IP (VoIP) introduction” on page 724) and IP Television (IPTV, see Chapter 21, “Internet Protocol Television” on page 745), are not suitable for such a network infrastructure. This is because WiFi clients using the same WAP must compete with each other for both bandwidth and attention from the WAP.

Conversely, WiMAX uses a scheduling algorithm that does guarantee QoS. Unlike the WiFi model, WiMAX clients must compete only for the initial entry into the network. After a client is granted entry, that client is guaranteed a time slot with the access point. Though the time slot might be expanded by the client based on need and availability, this initial guarantee lends itself to client applications that require a minimum QoS.

Other advantages WiMAX hold over WiFi include increased bandwidth (up to 70 Mbps), stronger encryption, and the ability to connect nodes that lack a line-of-site association. Additionally, as noted earlier, creating large WiFi hotspots requires the construction of multiple WAPs with overlapping smaller hotspots. WiMAX, however, is capable of servicing up to 30 miles (50 km) of service range. This makes WiMAX very suitable for rural areas, or remote areas in which installing the wiring to support any wired networks is cost-prohibitive.

Another application of WiMAX is to connect remote networks. Scenarios can exist when wired LANs or WiFi hotspots are preferred for a particular area. However, that area might be remote to other areas, and it is not cost-effective to connect the areas by WiFi or wires. Instead, these sites can be connected using WiMAX, thus bridging the distance between sites while still using the preferred network locally.

10.5 Applications of wireless networking

Given the benefits of wireless networking, there are several scenarios and problems to which wireless can be applied.

10.5.1 Last mile connectivity in broadband services

Last mile connectivity, sometimes called *last kilometer connectivity*, is a term commonly used by broadband providers (such as DSL or cable) to describe the final portion of the physical network used to provide network services. For example, this might be the wiring used to connect an individual home to a main cable. Installing the last mile often requires significant labor, high costs, and a lot of time. This is meaningful in respect to wireless because wireless presents a potential resolution to the last mile problem. The primary installation of the physical network can be attached to wireless radios, allowing subscribers to access network services without the installation of wiring.

10.5.2 Hotspots

A hotspot is any public location in which a wireless signal is present. These are often made available by businesses, such as coffee shops or restaurants, to provide Internet access to patrons. Note that some hotspots can be very large, such as those that span an university campus or an entire shopping mall. However, these are typically implementations of multiple overlapping hotspots that all broadcast the same SSID.

Hotspots can provide unlimited access to the Internet, or they can be restricted by the provider. Additionally, some commercial hotspots charge a fee before access to the Internet is granted. Many commercial hotspots include:

- A portal to which users are directed, allowing them to authenticate themselves or to pay a fee for Internet access.
- Some type of payment option, either directly to the establishment that maintains the hotspot, or through an Internet payment service.
- Free access to the internet, or limited access to prevent patrons from participating in illegal or questionable activities through the provider's hotspot.

10.5.3 Mesh networking

Mesh networking is a method of designing a network such that clients can act as repeaters, and repeaters can sometimes act as clients. In theory, this allows each node within a mesh network to be connected to every other node. Blocked routes can easily be bypassed, because a datagram can hop from node to node

until a new path is achieved. This essentially allows a meshed network to be self-healing.

This topology lends itself well in a wireless environment, because many nodes will be mobile, and therefore they will enter and leave the network continually.

10.6 IEEE standards relevant to this chapter

The following IEEE standards provide detailed information about the architecture and concepts presented in this chapter:

- 802.11 – Working Group for Wireless Local Area Networks. Reference:
  - 802.11a – Wireless LANs
  - 802.11b – Wireless Fidelity
  - 802.11g – 20+ Mbps Wireless connectivity
- 802.16 – Working Group for Wireless Metropolitan Area Networks. Reference:
  - 802.16a – Radio Spectrum Use
  - 802.16b – Five to six GHz Spectrum Use, Quality of Service
  - 802.16c – Ten to sixty-six GHz Spectrum Use
  - 802.16e – Mobile WiMax

Archived

Archived

Part 2

TCP/IP application protocols

Included in the TCP/IP suite of protocols is an extensive list of applications designed to make use of the suite's services. It is through these entities that resources can be made available, data can be moved between hosts, and remote users can communicate. Examples of applications architected within the TCP/IP suite include the File Transfer Protocol (FTP) and the Simple Mail Transport Protocol (SMTP). Other applications have been architected to manage networks and provide seamless access to resources. These include applications such as the Domain Name System (DNS) and the Simple Network Management Protocol (SNMP).

However, applications that make use of TCP/IP services are not limited to RFC architected protocols defined in parallel to TCP/IP. Other proprietary and open-source applications exist, defined either by industry standards or by open-organization specifications. Some of these applications, such as sendmail and the Common Internet File System (CIFS), mimic the services offered by RFC architected protocols. Others, however, fulfill specific needs not specifically addressed by RFCs. An example of the latter is the Wireless Application

© Copyright IBM Corp. 1989-2006. All rights reserved.

Protocol, which is defined by the Open Mobile Alliance (OMA) and is defined in specifications created by that organization. These OMA specifications are available at:

http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

Archived

11

Application structure and programming interfaces

Application protocols consist of the highest level of protocols in the OSI model. These protocols act as user interfaces to the TCP/IP protocol suite. In this chapter, we discuss the following topics:

- Characteristics of applications
  - The client/server model
- Application programming interfaces (APIs)
  - The socket API
  - Remote Procedure Call (RPC)
  - The SNMP distributed programming interface (SNMP DPI)
  - REXX sockets
- Related RFCs

© Copyright IBM Corp. 1989-2006. All rights reserved.

11.1 Characteristics of applications

Each of the application protocols share come common characteristics:

- They can be user-written applications or applications standardized and shipped with the TCP/IP product. Examples of applications native to the TCP/IP protocol suite include:
  - Telnet, which provides interactive terminal access to remote hosts
  - The File Transfer Protocol (FTP), which provides the ability to transfer files between remote hosts
  - The Simple Mail Transfer Protocol (SMTP), which provides an Internet mailing system

  While these are widely implemented application protocols, many others exist.
- They use either UDP or TCP as a transport mechanism. Remember that UDP (see 4.2, “User Datagram Protocol (UDP)” on page 146) is unreliable and offers no flow control. In this case, the application must provide its own error recovery and flow control routines. For this reason, it is often easier to build applications that use TCP (see 4.3, “Transmission Control Protocol (TCP)” on page 149), a reliable, connection-oriented protocol.
- Most applications implement the client/server model of interaction.

11.1.1 The client/server model

TCP is a peer-to-peer, connection-oriented protocol. There are no master/subordinate relationships, in which one instance of the application protocol controls or is controlled by another instance. Instead, the applications use a client/server model for communications. In such a model, the server offers a service to users. The client is the interface by which the user accesses the offered service. Both a client instance and a server instance must be active for the application protocol to operate. Note that the both instances can reside on the same host or on different hosts (see Figure 11-1 on page 409).

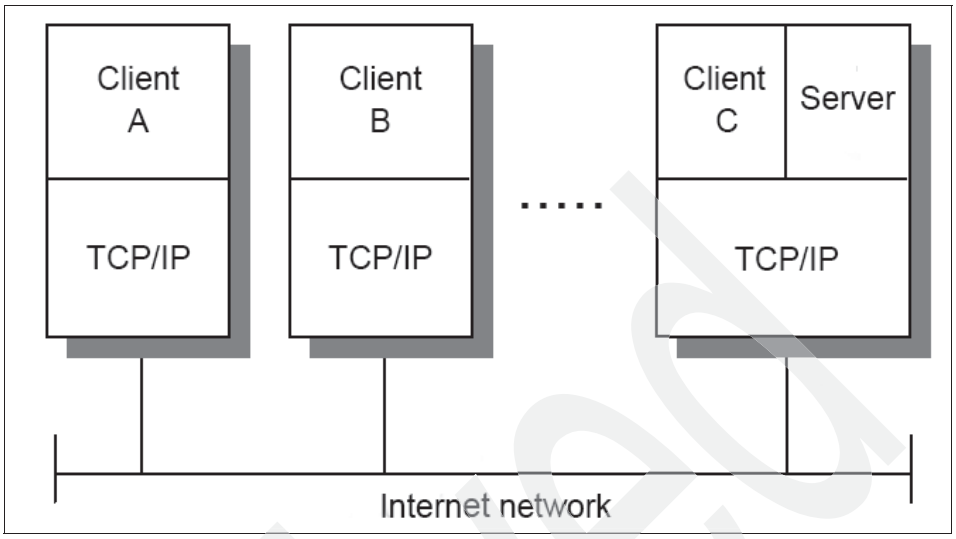


Figure 11-1 The client/server model of applications

In the previous figure, client A and client B represent client instances on remote hosts. Client C represents a client instance on the same system as the server instance. Through the client, a user can generate a request for the service provided by the server. The request is then delivered to the server using TCP/IP as the transport vehicle.

Upon receiving the request, the server performs the desired service, and then sends a reply back to the client. A server typically can accept and process multiple requests (multiple clients) at the same time.

Common servers, such as Telnet, FTP, and SMTP, listen for requests on well-known ports (see 4.1.1, “Ports” on page 144). This allows a client to connect to the server without having determine on what port the server is listening. Clients that need to connect to a nonstandard server application, or to a standard server application that has been configured to listen on a port other than the well-known port, must implement another mechanism to determine on which port a server is listening. This mechanism might employ a registration service, such as portmap or Remote Procedure Call Bind (RPCBIND), to identify the port to which a request should be sent. Both portmap and RPCBIND are defined by RFC 1833.

11.2 Application programming interfaces (APIs)

An application programming interface (API) enables developers to write applications that can make use of TCP/IP services. The following sections provide an overview of the most common APIs for TCP/IP applications.

11.2.1 The socket API

The socket interface is one of several APIs to the communication protocols. Designed to be a generic communication programming interface, it was first introduced by the 4.2BSD UNIX-based system. Although the socket API for IPv4 was never standardized, it has become a de facto industry standard, and RFC 3493 was created to update the API for IPv6. More advanced IPv6 socket programming can be found in RFC 3542.

The socket interface is differentiated by the following services provided to applications:

- Stream sockets services

  Stream sockets provide a reliable connection-oriented service such as TCP. Data is sent without errors or duplication, and is received in the same order as it is sent. Flow control is built in to avoid data overruns. No boundaries are imposed on the exchanged data, which is considered a stream of bytes. An example of an application that uses stream sockets is the File Transfer Protocol (FTP).

- Datagram sockets services

  Datagram sockets define a connectionless service such as UDP. Datagrams are sent as independent packets. The service does not guarantee successful delivery of the packets; data can be lost or duplicated, and datagrams can arrive out of order. No disassembly and reassembly of packets is performed. An example of an application that uses datagram sockets is the Network File System (NFS).

- Raw sockets services

  Raw sockets allow direct access to lower layer protocols, such as IP and ICMP. This interface is often used for testing new protocol implementations. An example of an application that uses raw sockets is the **`ping`** command.

Additional information about sockets is in 4.1.2, “Sockets” on page 145. Socket APIs provide functions that enable applications to perform the following actions:

- Initialize a socket
- Bind (register) a socket to a port address

Archived

- Listen on a socket for inbound connections
- Accept an inbound connection
- Connect outbound to a server
- Send and receive data on a socket
- Close a socket

Though the specific details of the previous functions will vary from platform to platform, the industry standard is based on Berkeley sockets, also known as the BSD socket API, released in 1989. Additionally, RFC 3493 was created to define the extensions needed for socket APIs to incorporate IPv6. The core functions made available by industry standard APIs are as follows:

- Initialize a socket

  Format:

  ```
  socket(domain, type, protocol)
  ```

  Definitions of fields:

  domain This is the protocol family of the socket to be created. Valid values include PF_INET (IPv4) and PF_INET6 (IPv6). Additional platform-specific values can also be used.

  type This is the type of socket to be opened. Valid values typically include stream, datagram, and raw.

  protocol This is the protocol that will be used on the socket. Values typically include UDP, TCP, and ICMP.

- Bind a socket to a port address

  Format:

  ```
  bind(sockfd, localaddress, addresslength)
  ```

  Definition of fields:

  sockfd This is the socket that is to be bound to the port address. This is the value obtained previously from the **socket** function.

  localaddress This is the socket address structure to which the socket is bound.

  addresslength This is the length of the socket address structure.

- Listen on a socket for inbound connections

  Format:

  ```
  listen(sockfd, queuesize)
  ```


Archive

Definition of fields:

sockfd This is the socket on which the application is to listen. This is the value obtained previously from the **socket** function.

queuesize This is the number of inbound requests that can be queued by the system at any single time.

Note: The **listen()** function is typically invoked by server applications. The function is called to await inbound connections from clients.

- Accept an inbound connection

  Format:

  ```
  accept(sockfd, remoteaddress, addresslength)
  ```

  Definition of fields:

  sockfd This is the socket on which the connection is to be accepted. This is the value obtained previously from the **socket** function.

  remoteaddress This is the remote socket address structure from which the connection was initiated.

  addresslength This is the length of the socket address structure.

  Note: The **accept()** function is typically invoked by server applications to accept connections from clients. The remote address is a place holder in which the remote address structure will be stored.

- Connect outbound to a server

  Format:

  ```
  connect(sockfd, remoteaddress, addresslength)
  ```

  Definition of fields:

  sockfd This is the socket from which the connection is to be opened. This is the value obtained previously from the **socket** function.

  remoteaddress This is the remote socket address structure to which the connection is to be opened.

  addresslength This is the length of the socket address structure.

  Note: The **connect** function is typically invoked by client applications.

Archived

- Send and receive data on a socket

  Format:

  ```
  sendmsg(sockfd, data, datalength, flags)
  recvmsg(sockfd, data, datalength, flags)
  ```

  Definition of fields:

  sockfd This is the socket across which the data will be sent or read.

  data This is the data to be sent, or the buffer into which the read data will be placed.

  datalength When writing data, this is the length of the data to be written. When reading data, this is the amount of data to be read from the socket.

  flags This field, which is in many implementations optional, provides any specific information to TCP/IP regarding any special actions to be taken on the socket when sending or receiving the data.

  Note: Other variations of **sendmsg()** and **recv()** can be as follows:

  ```
  sendmsg(): send(), sendto(), write()
  recvmsg(): recv(), recvfrom(), read()
  ```

  RFC 3493 does not specifically discuss the fields passed on the **sendmsg()** function. The fields discussed earlier are drawn from those typically used by most implementations.

- Close a socket

  Format:

  ```
  close(sockfd)
  ```

  Definition of fields:

  sockfd This is the socket which is to be closed.

Archived

An example of a client/server scenario

Figure 11-2 illustrates the appropriate order of socket API functions to implement a connection-oriented interaction.

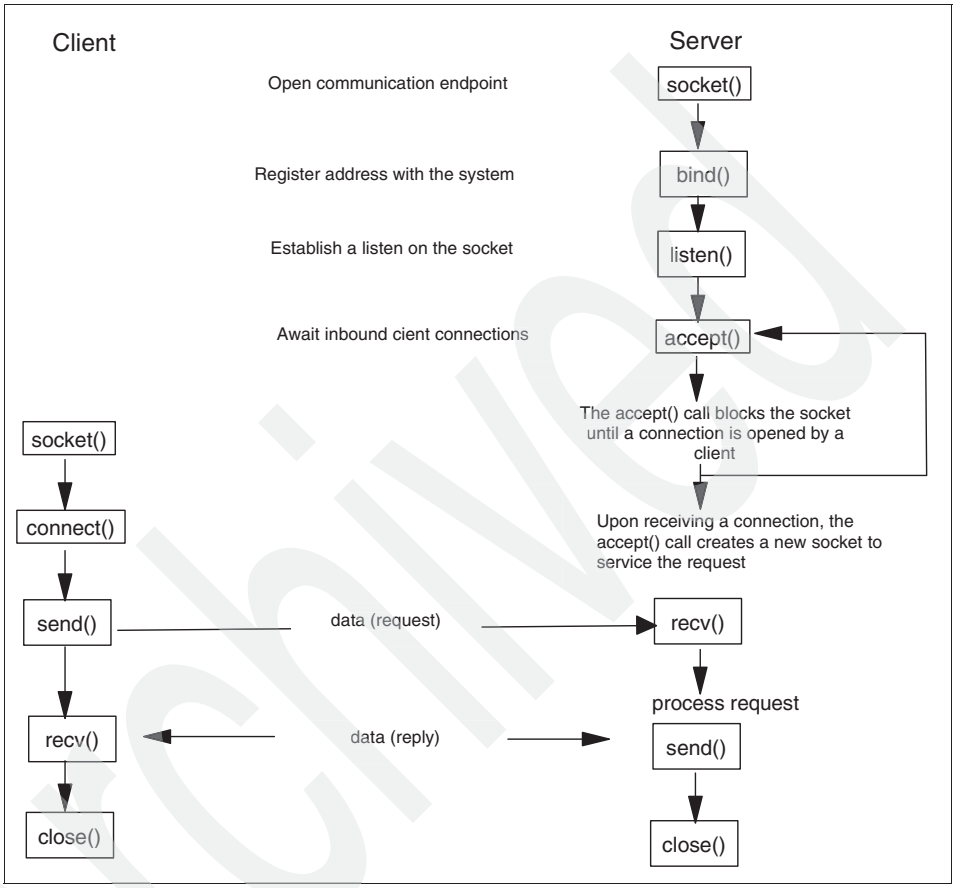


Figure 11-2 Socket system calls for a connection-oriented protocol

Archive

The connectionless scenario is simpler in that the **`listen()`**, **`accept()`**, and **`connect()`** functions are not invoked. Table 11-1 compares the socket API functions that are used for connection-oriented and connectionless clients and servers.

Table 11-1 Socket API function comparison

Client/server connection	Establish	Send	Receive

Connection-oriented server	`bind()`
	`listen()`
	`accept()`
		`send()`
		`sendto()`
		`write()`
			`recv()`
			`recvfrom()`
			`read()`

Connection-oriented client	`connect()`	`send()`
		`sendto()`
		`write()`
			`recv()`
			`recvfrom()`
			`read()`

Connectionless server	`bind()`	`sendto()`	`recvfrom()`
Connectionless client	`bind()`	`sendto()`	`recvfrom()`

11.2.2 Remote Procedure Call (RPC)

Remote Procedure Call (RPC), originally developed by Sun Microsystems and currently used by many UNIX-based systems, is an application programming interface (API) available for developing distributed applications. It allows programs to execute subroutines on a remote system. The caller program, which represents the client instance in the client/server model (see Figure 11-1 on page 409), sends a call message to the server process and waits for a reply message. The call message includes the subroutine's parameters, and the reply message contains the results of executing the subroutine. RPC also provides a standard way of encoding data passed between the client/server in a portable fashion called External Data Representation (XDR), defined by RFC 4506/STD 0067.

The RPC concept

The concept of RPC is very similar to that of an application program issuing a procedure call:

1. The caller process, which represents the client instance in the client/server model (see Figure 11-1 on page 409,) sends a call message and waits for the reply.
2. The server awaits the arrival of call messages. When a call message arrives, the server process extracts the procedure parameters, computes the results, and sends them back in a reply message.

Figure 11-3 shows the conceptual model of RPC.

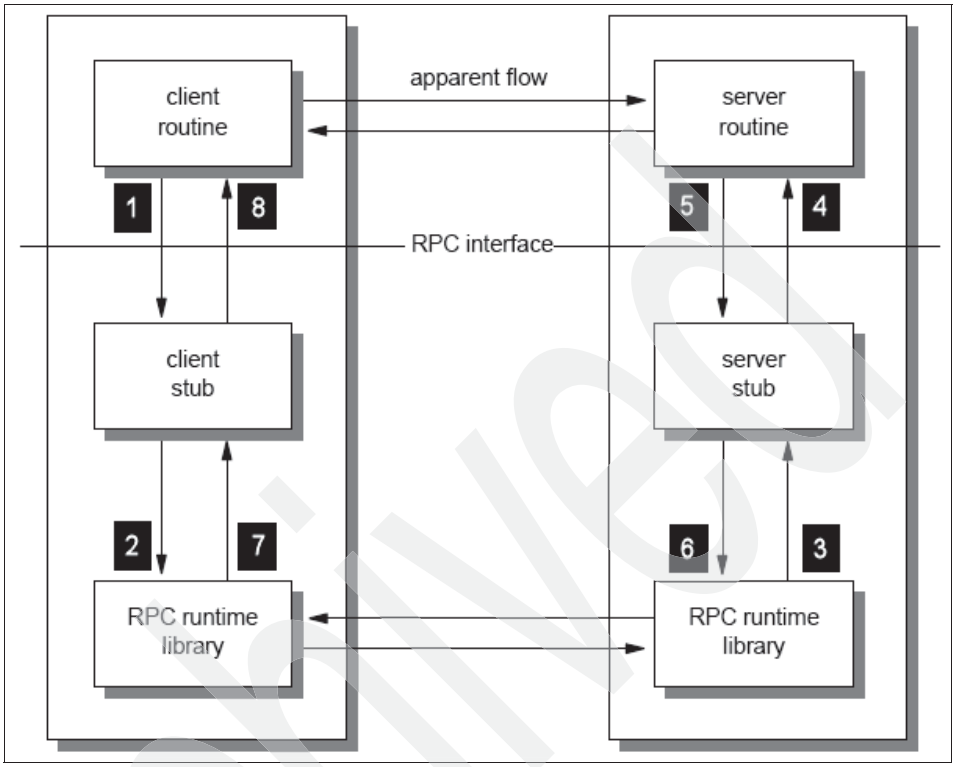


Figure 11-3 Remote Procedure Call (RPC) conceptual model

Figure 11-3 represents only one possible model. In this figure, the caller's execution blocks until a reply message is received. Other models are possible; for example, the caller can continue processing while waiting for a reply, or the server can dispatch a separate task for each incoming call so that it remains free to receive other messages.

Note: The client and server stub seen in Figure 11-3 are created by the RPCGEN command, discussed later in this chapter.

The Remote Procedure Calls differ from local procedure calls in the following ways:

- The use of global variables is not possible because the server has no access to the caller program's address space.
- Performance might be affected by the transmission times.

- User authentication might be necessary.
- The location of the server must be known.

Call message and reply transport

The RPC protocol can be implemented on any transport protocol. In the case of TCP/IP, it can use either TCP or UDP as the transport vehicle. When using UDP, remember that this does not provide reliability. Therefore, it is the responsibility of the caller program to employ any needed reliability (using timeouts and retransmissions, usually implemented in RPC library routines). Note that even with TCP, the caller program still needs a timeout routine to deal with exceptional situations, such as a server crash or poor network performance.

RPC call message structure

The RPC call message consists of several fields:

- Program and procedure numbers

  Each call message contains three unsigned integer fields that uniquely identify the procedure to be executed:

  - Remote program number
  - Remote program version number
  - Remote procedure number

  The remote program number identifies a functional group of procedures, for example, a file system, that includes individual procedures such as read and write. These individual procedures are identified by a unique procedure number within the remote program. As the remote program evolves, a version number is assigned to the different releases.

  Each remote program is attached to an internet port. The number of this port can be freely chosen, except for the reserved well-known-services port numbers. It is evident that the caller will have to know the port number used by this remote program.

- Authentication fields

  Two fields, credentials and verifier, are provided for the authentication of the caller to the service. It is up to the server to use this information for user authentication. Also, each implementation is free to choose the varieties of supported authentication protocols. These authentication protocols include:

  - Null authentication.
  - UNIX authentication: The callers of a remote procedure can identify themselves as they are identified on the UNIX system.

  - DES authentication: In addition to a user ID, a time stamp field is sent to the server. This time stamp is the current time, enciphered using a key known to the caller machine and server machine only (based on the secret key and public key concept of DES).

- Procedure parameters

  Procedure parameters consist of the data passed to the remote procedure.

RPC message reply structure

Several replies exist, depending on the action taken:

- SUCCESS: Procedure results are sent back to the client.
- RPC_MISMATCH: Server is running another version of RPC than the caller.
- AUTH_ERROR: Caller authentication failed.
- PROG_MISMATCH: If a program is unavailable, or if the version asked for does not exist, or if the procedure is unavailable.

For a detailed description of the call and reply messages, see RFC 1057 – RPC: Remote Procedure Call Protocol Specification Version 2, which also contains the type definitions (typedef) for the messages in XDR language.

Using portmap to register a program's port number

The portmap (or portmapper) is a server application that maps a program and its version number to the Internet port number used by the program. Portmap is assigned the reserved (well-known service) port number 111.

Portmap only knows about RPC programs for the host on which portmap runs. In order for portmap to know about these programs, each program should, when initializing, register itself with the local portmap application.

The RPC client (caller) has to ask the portmap service on the remote host about the port used by the desired server program. Normally, the calling application contacts portmap on the destination host to obtain the correct port number for a particular remote program, and then sends the call message to this particular port. A variation can exist such that the caller can also send the procedure data along to portmap, and then portmap directly invokes the desired procedure using the received data. Figure 11-4 on page 419 illustrates this process.

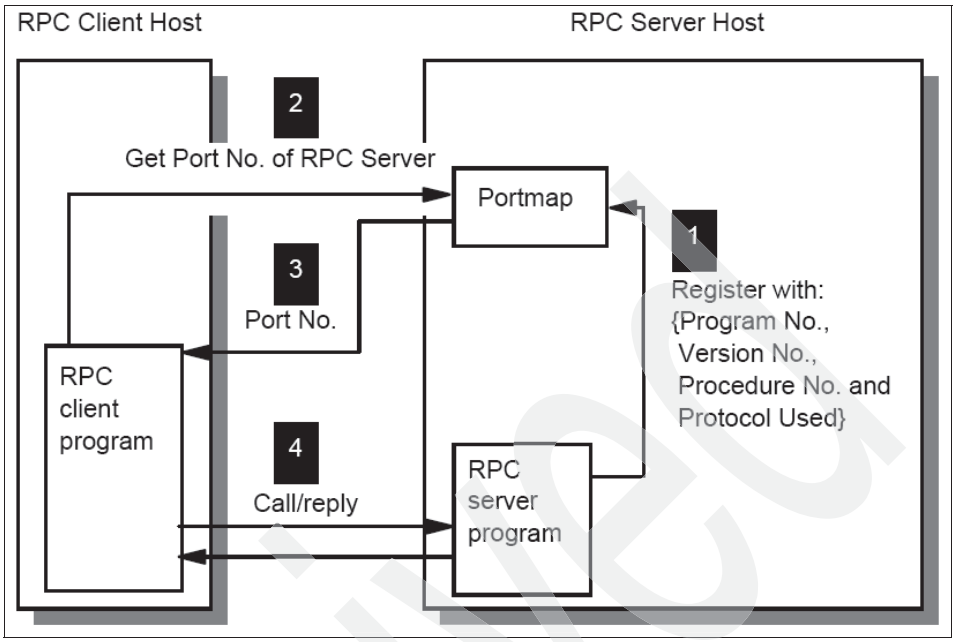


Figure 11-4 RPC using portmap

RPCGEN

RPCGEN is a tool that generates C code to implement an RPC protocol. The input to RPCGEN is a file written in a language similar to C, known as the RPC language. The output of the tool is four files used to implement the RPC program. For example, assume that RPCGEN is used with an input file named proto.x. The output will consist of the following files:

- proto.h: A header file containing common definitions of constants and macros used by the RPC program
- protoc.c: The client stub source file, shown in Figure 11-3 on page 416
- protos.c: The server stub source file, also shown in Figure 11-3 on page 416
- protox.c: The XDR routines source file

11.2.3 The SNMP distributed programming interface (SNMP DPI)

The Simple Network Management Protocol (SNMP) defines a protocol that permits operations on a collection of system-specific variables, also called objects. This set of variables, called the *Management Information Base* (MIB), and a core set of variables have previously been defined. However, the design of the MIB makes provision for extension of this core set. However, conventional

SNMP agent implementations provide no means for a user to make new objects available. The SNMP DPI addresses this issue by providing a lightweight mechanism that permits users to dynamically add, delete, or replace application-specific or user-specific management variables in the local MIB without requiring recompilation of the SNMP agent. This is achieved by writing a subagent that communicates with the agent through the SNMP distributed programming interface (DPI), described in RFC 1592.

The SNMP DPI allows a process to register the existence of a MIB object with the SNMP agent. When requests for this object are received by the SNMP agent, it will pass the query on to the process acting as a subagent. This subagent then returns an appropriate answer to the SNMP agent, which then packages an SNMP response packet and sends it to the remote network management station that initiated the request.

The DPI communication between the SNMP agent and the subagent is transparent to the remote network management stations. This communication can occur over a variety of transports, including stream, datagram, and UNIX sockets.

Using the SNMP DPI, a subagent can:

- Create and delete subtrees in the application-specific or user-specific MIB
- Create a register request packet to inform the SNMP agent of the MIB supported by the subagent
- Create response packet to answer requests received from the SNMP agent
- Create a TRAP request packet to be delivered by the SNMP agent to remote management stations

Archive

The interaction between a subagent and the SNMP agent is illustrated in Figure 11-5.

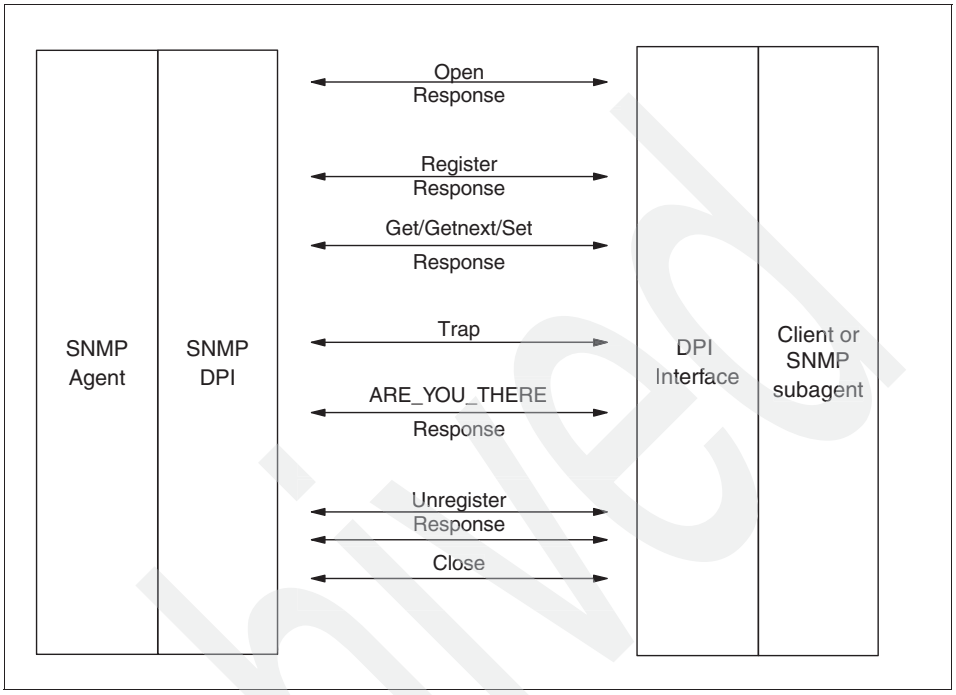


Figure 11-5 SNMP DPI overview

An SNMP subagent, running as a separate process (potentially on another machine), can set up a connection with the agent. The subagent has an option to communicate with the SNMP agent through UDP, TCP sockets, or other mechanisms.

After the connection is established, the subagent issues a DPI OPEN and one or more REGISTER requests to register one or more MIB subtrees with the SNMP agent. The SNMP agent responds to DPI OPEN and REGISTER requests with a RESPONSE packet, indicating success or failure.

As the SNMP agent receives queries from remote management stations, any requests for an object in a subtree registered by a subagent are forwarded through the DPI to that subagent. These requests are in the form of GET, GENEXT, or SET.

Archived

Note: SET requests do not conform to the request/response sequence used by GET or GETNEXT requests. Instead, SET requests occur in a sequence of SET/COMMIT, SET/UNDO, or SET/COMMIT/UNDO. We discuss this in greater detail in Chapter 17, “Network management” on page 623.

The subagent sends responses back to the SNMP agent through a RESPONSE packet, which the SNMP agent then encodes into an SNMP packet sent back to the requesting remote management station.

If the subagent wants to report an important state change, it can send a DPI TRAP packet to the SNMP agent, which the agent will encode into an SNMP trap packet and send to designated remote management stations.

A subagent can send an ARE_YOU_THERE to verify that the connection is still open. If so, the agent sends a RESPONSE with no error; otherwise, it sends a RESPONSE with an error.

If the subagent wants to stop operations, it sends a DPI UNREGISTER and a DPI CLOSE packet to the agent. The agent sends a response to an UNREGISTER request, but there is no RESPONSE to a CLOSE. A CLOSE implies an UNREGISTER for all registrations that exist for the DPI connection being CLOSED, even if no UNREGISTER has been sent.

An agent can send a DPI UNREGISTER if a higher priority registration arrives (or for other reasons) to the subagent. The subagent then responds with a DPI RESPONSE packet. An agent can also send a DPI CLOSE to indicate that it is terminating the DPI connection. This might occur when the agent terminates, or if the agent has timed out while awaiting a response from a subagent.

11.2.4 REXX sockets

The Restructured Extended Executor (REXX) programming language was originally developed by IBM in 1982, but since then a wide variety of platforms have created implementations for their platforms. In order to enable REXX applications to communicate over the TCP/IP network, most platforms have created a REXX socket API to allow:

- Socket initialization
- Data exchange through sockets
- Management activity through sockets
- Socket termination

However, currently there is no standard to define the REXX socket API. The specifics of the functions provided vary from platform to platform.

11.3 RFCs relevant to this chapter

The following RFCs contain detailed information about application structure and programming interfaces:

- RPC 1057 – Remote Procedure Call Protocol specification: Version 2 (June 1988)
- RFC 1833 – Binding Protocols for ONC™ RPC Version 2 (August 1995)
- RFC 1592 – Simple Network Management Protocol Distributed Protocol Interface Version 2.0 (March 1994)
- RFC 3493 – Basic Socket Interface Extensions for IPv6 (February 2003)
- RFC 3542 – Advanced Sockets Application Program Interface (API) for IPv6 (May 2003)
- RFC 4506 – XDR: External Data Representation Standard. (Also STD0067) (May 2006)

Archived

Archived

12

Directory and naming protocols

An inherent problem in making resources available through a network is creating an easy way of accessing these resources. On small networks, it might be easy enough to simply remember or write down the information needed to remotely access resources. However, this solution does not scale as networks, and the number of available resources, continue to grow. This becomes increasingly more complex as resources are made available outside of individual networks, to multiple networks, or even across the Internet, on multiple platforms and across a variety of differing hardware. To overcome this, directory and naming methods were devised to provide a uniform method of obtaining the information needed to access a networked resource. This chapter reviews four of these methods:

- Domain Name System (DNS)
- Dynamic Domain Name System (DDNS)
- Network Information System (NIS)
- Lightweight Directory Access Protocol (LDAP)

© Copyright IBM Corp. 1989-2006. All rights reserved.

12.1 Domain Name System (DNS)

The Domain Name System is a standard protocol with STD number 13, and its status is recommended. It is described in RFC 1034 and RFC 1035. This section explains the implementation of the Domain Name System and the implementation of name servers.

The early Internet configurations required the use of only numeric IP addresses. Because this was burdensome and much harder to remember than just the name of a system, this evolved into the use of symbolic host names. For example, instead of typing:

```
TELNET 10.12.7.14
```

You can type:

```
TELNET MyHost
```

MyHost is then translated in some way to the IP address 10.12.7.14. Though using host names makes the process of accessing a resource easier, it also introduces the problem of maintaining the mappings between IP addresses and high-level machine names in a coordinated and centralized way.

Initially, host names to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT), which was fetched by all hosts using FTP. This is called a *flat namespace*. But due to the explosive growth in the number of hosts, this mechanism became too cumbersome (consider the work involved in the addition of just one host to the Internet) and was replaced by a new concept: *Domain Name System*. Hosts on smaller networks can continue to use a local flat namespace (the HOSTS.LOCAL file) instead of or in addition to the Domain Name System. Outside of small networks, however, the Domain Name System is essential. This system allows a program running on a host to perform the mapping of a high-level symbolic name to an IP address for any other host without requiring every host to have a complete database of host names.

12.1.1 The hierarchical namespace

Consider the typical internal structure of a large organization. Because the chief executive cannot do everything, the organization will probably be partitioned into divisions, each of them having autonomy within certain limits. Specifically, the executive in charge of a division has authority to make direct decisions, without permission from the chief executive.

Domain names are formed in a similar way, and will often reflect the hierarchical delegation of authority used to assign them. For example, consider the name:

```
myHost.myDept.myDiv.myCorp.com
```

In this example, we know that there is a single host name *myHost*, which exists within the *myDept.myDiv.myCorp* subdomain. The *myDept.myDiv.myCorp* subdomain is one of the subdomains of *myDiv.myCorp.com* subdomain, which is in turn one of the subdomains of *myCorp.com*. Finally, *myCorp.com* is a subdomain of *com*. This hierarchy is better illustrated in Figure 12-1.

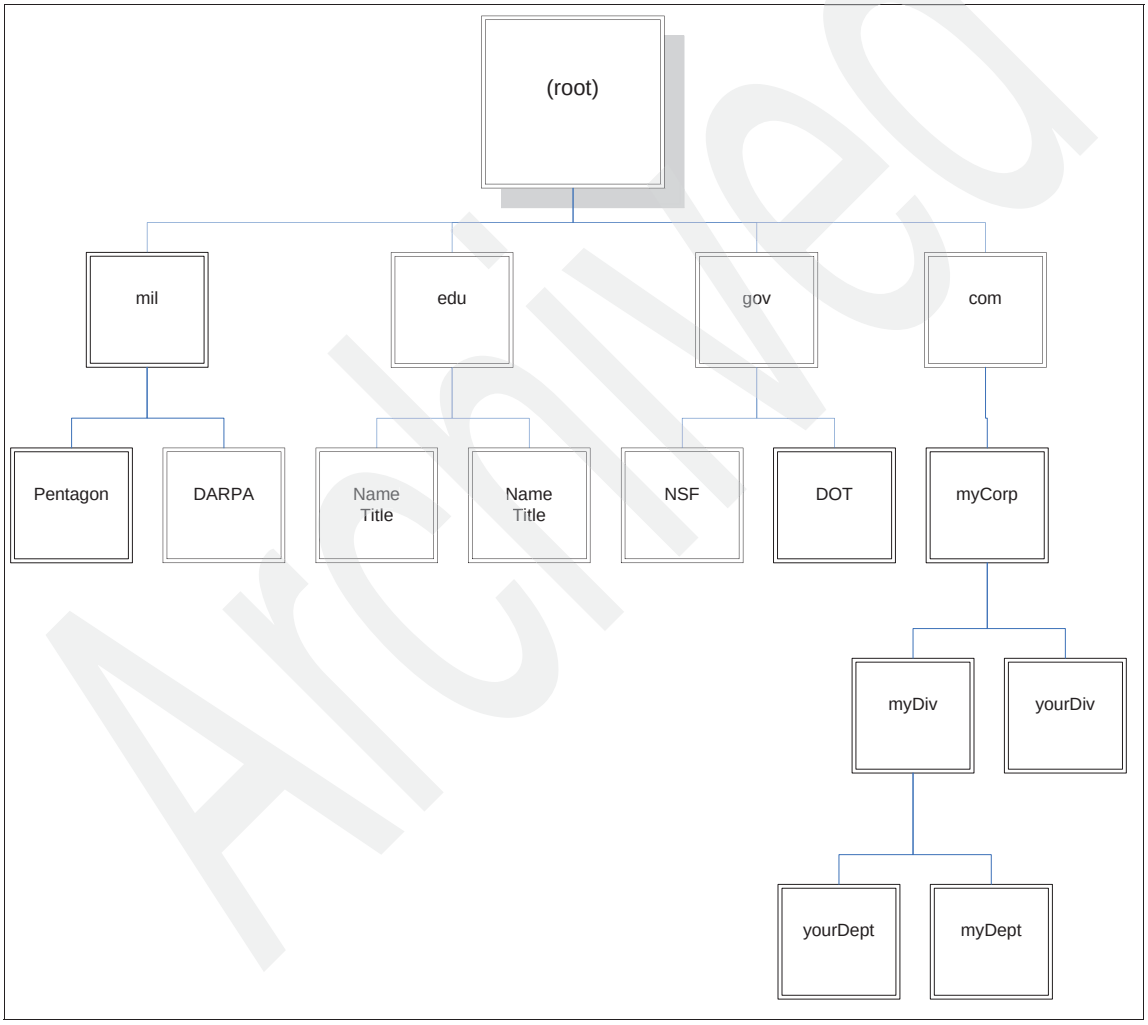


Figure 12-1 DNS hierarchical namespace

Archived

We discuss this hierarchical structure at greater length in the following sections.

12.1.2 Fully qualified domain names (FQDNs)

When using the Domain Name System, it is common to work with only a part of the domain hierarchy, such as the myDivision.myCorp.com domain. The Domain Name System provides a simple method of minimizing the typing necessary in this circumstance. If a domain name ends in a dot (for example, myDept.myDiv.myCorp.com.), it is assumed to be complete. This is called a *fully qualified domain name (FQDN)* or an *absolute domain name*. However, if it does not end in a dot (for example, myDept.myDiv), it is incomplete and the DNS resolver may complete this by appending a suffix such as .myCorp.com to the domain name. The rules for doing this are implementation-dependent and locally configurable.

12.1.3 Generic domains

The top-level names are called the generic top-level domains (gTLDs), and can be three characters or more in length. Table 12-1 shows some of the top-level domains of today's Internet domain namespace.

Table 12-1 Current generic domains

Domain name	Meaning
aero	The air transport industry
biz	Business use
cat	The Catalan culture
com	Commercial organizations
coop	Cooperatives
edu	Educational organizations
gov	U.S. governmental agencies
info	Informational sites
int	International organizations
jobs	Employment-related sites
mil	The U.S. military
mobi	Mobile devices sites
museum	Museums

Domain name	Meaning
name	Family and individual sites
net	Network infrastructures
org	Non-commercial organizations
pro	Professional sites
travel	The travel industry

These names are registered with and maintained by the Internet Corporation for Assigned Names and Numbers (ICANN). For current information, see the ICANN Web site at:

`http://www.icann.org`

12.1.4 Country domains

There are also top-level domains named for the each of the ISO 3166 international 2-character country codes (from *ae* for the United Arab Emirates to *zw* for Zimbabwe). These are called the *country* domains or the *geographical* domains. Many countries have their own second-level domains underneath which parallel the generic top-level domains. For example, in the United Kingdom, the domains equivalent to the generic domains *.com* and *.edu* are *.co.uk* and *.ac.uk* (ac is an abbreviation for academic). There is a .us top-level domain, which is organized geographically by state (for example, .ny.us refers to the state of New York). See RFC 1480 for a detailed description of the .us domain.

12.1.5 Mapping domain names to IP addresses

The mapping of names to addresses consists of independent, cooperative systems called name servers. A name server is a server program that holds a master or a copy of a name-to-address mapping database, or otherwise points to a server that does, and that answers requests from the client software, called a name resolver.

Conceptually, all Internet domain servers are arranged in a tree structure that corresponds to the naming hierarchy in Figure 12-1 on page 427. Each leaf represents a name server that handles names for a single subdomain. Links in the conceptual tree do not indicate physical connections. Instead, they show which other name server a given server can contact.

12.1.6 Mapping IP addresses to domain names: Pointer queries

The Domain Name System provides for a mapping of symbolic names to IP addresses and vice versa. While the hierarchical structure makes it easy in principle to search the database for an IP address using its symbolic name, the process of mapping an IP address to a symbolic name cannot use the same process. Therefore, there is another namespace that facilitates the reverse mapping of IP address to symbolic name. It is found in the domain in-addr.arpa (arpa is used because the Internet was originally the *ARPAnet*).

Not including IPv6, IP addresses are normally written in dotted decimal format, and there is one layer of domain for each hierarchy. Contrary to domain names, which have the least-significant parts of the name first, the dotted decimal format has the most significant bytes first. Therefore, in the Domain Name System, the dotted decimal address is shown in reverse order.

For example, consider the following IPv4 address:

```
129.34.139.30
```

The in-add.arpa address for this is:

```
30.139.34.129.in-addr.arpa.
```

This is handled slightly different for IPv6 addresses. Because of the IPv6 address' structure, the reverse order is done in nibbles in stead of octets. Also, the in-addr.arpa domain does not include IPv6. Instead, the domain used is IP6.ARPA. For example, consider the following IPv6 address:

```
4321:0:1:2:3:4:567:89ab
```

Breaking this into nibbles, reversing the odder, and appending the domain yields:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA
```

Given an IP address, the Domain Name System can be used to find the matching host name. A domain name query to do this is called a *pointer query*.

12.1.7 The distributed name space

The Domain Name System uses the concept of a *distributed name space*. Symbolic names are grouped into *zones of authority*, more commonly referred to as *zones*. In each of these zones, one or more hosts has the task of maintaining a database of symbolic names and IP addresses within that zone, and provides a server function for clients who want to translate between symbolic names and IP addresses. These local name servers are then (through the internetwork on which they are connected) logically interconnected into a hierarchical tree of *domains*. Each zone contains a part or a *subtree* of the hierarchical tree, and the

names within the zone are administered independently of names in other zones. Authority over zones is vested in the name servers.

Normally, the name servers that have authority for a zone will have domain names belonging to that zone, but this is not required. Where a domain contains a subtree that falls in a different zone, the name server or servers with authority over the superior domain are said to *delegate authority* to the name server or servers with authority over the subdomain. Name servers can also delegate authority to themselves; in this case, the domain name space is still divided into zones moving down the domain name tree, but authority for two zones is held by the same server. The division of the domain name space into zones is accomplished using resource records stored in the Domain Name System.

At the top-level root domain there is an exception to this. There is no higher system to which authority can be delegated, but it is not desirable to have all queries for fully qualified domain names to be directed to just one system. Therefore, authority for the top-level zones is shared among a set of *root name servers*[1] coordinated by the ICANN.

To better illustrate the process of resolving a symbolic name to an IP address, consider a query for myHost.myDept.myCorp.com, and let us assume that our name server does not have the answer already in its cache. The query goes to the .com root name server, which in turn forwards the query to a server with an NS record for myCorp.com. At this stage, it is likely that a name server has been reached that has cached the needed answer. However, the query could be further delegated to a name server for myDept.myCorp.com

As a result of this scheme:

- Rather than having a central server for the database, the work that is involved in maintaining this database is off-loaded to hosts throughout the name space.
- Authority for creating and changing symbolic host names and responsibility for maintaining a database for them is delegated to the organization owning the zone (within the name space) containing those host names.
- From the user's standpoint, there is a single database that deals with these address resolutions. The user might be aware that the database is distributed, but generally need not be concerned about this.

[1] At the time of writing, there were 13 root servers.

Note: Although domains within the namespace will frequently map in a logical fashion to networks and subnets within the IP addressing scheme, this is not a requirement of the Domain Name System. Consider a router between two subnets. It has two IP addresses, one for each network adapter, but it would not normally have two symbolic names.

12.1.8 Domain name resolution

The domain name resolution process can be summarized in the following steps:

1. A user program issues a request such as the **gethostbyname()** system call (this particular call asks for the IP address of a host by passing the host name) or the **gethostname()** system call (which asks for a host name of a host by passing the IP address).
2. The resolver formulates a query to the name server. (Full resolvers have a local name cache to consult first; stub resolvers do not. See "Domain name full resolver" and "Domain name stub resolver" on page 434.)
3. The name server checks to see if the answer is in its local authoritative database or cache, and if so, returns it to the client. Otherwise, it queries other available name servers, starting down from the root of the DNS tree or as high up the tree as possible.
4. The user program is finally given a corresponding IP address (or host name, depending on the query) or an error if the query could not be answered. Normally, the program will not be given a list of all the name servers that have been consulted to process the query.

Domain name resolution is a client/server process (see 11.1.1, "The client/server model" on page 408). The client function (called the *resolver* or *name resolver*) is transparent to the user and is called by an application to resolve symbolic high-level names into real IP addresses or vice versa. The name server (also called a *domain name server*) is the server application providing the translation between high-level machine names and the IP addresses. The query/reply messages can be transported by either UDP or TCP.

Domain name full resolver

Figure 12-2 shows a program called a *full resolver*, which is distinct from the user program, that forwards all queries to a name server for processing. Responses are cached by the name server for future use.

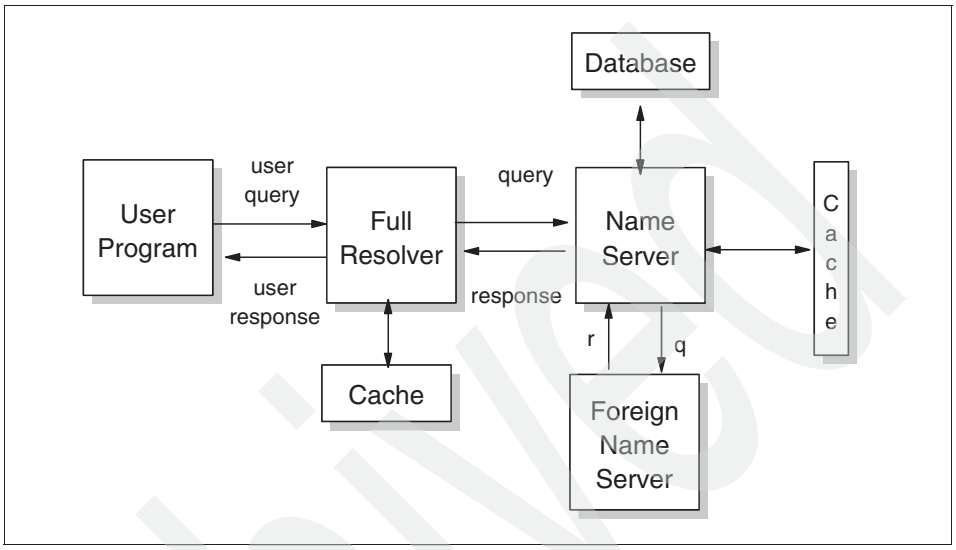


Figure 12-2 DNS: Using a full resolver for domain name resolution

Archived

Domain name stub resolver

Figure 12-3 shows a *stub resolver*, a routine linked with the user program, that forwards the queries to a name server for processing. Responses are cached by the name server, but not usually by the resolver, although this is implementation dependent. On most platforms, the stub resolver is implemented by two library routines (or by some variation of these routines): **`gethostbyname()`** and **`gethostbyaddr()`**. These are used for converting host names to IP addresses and vice versa. Stub resolvers are much more common than full resolvers.

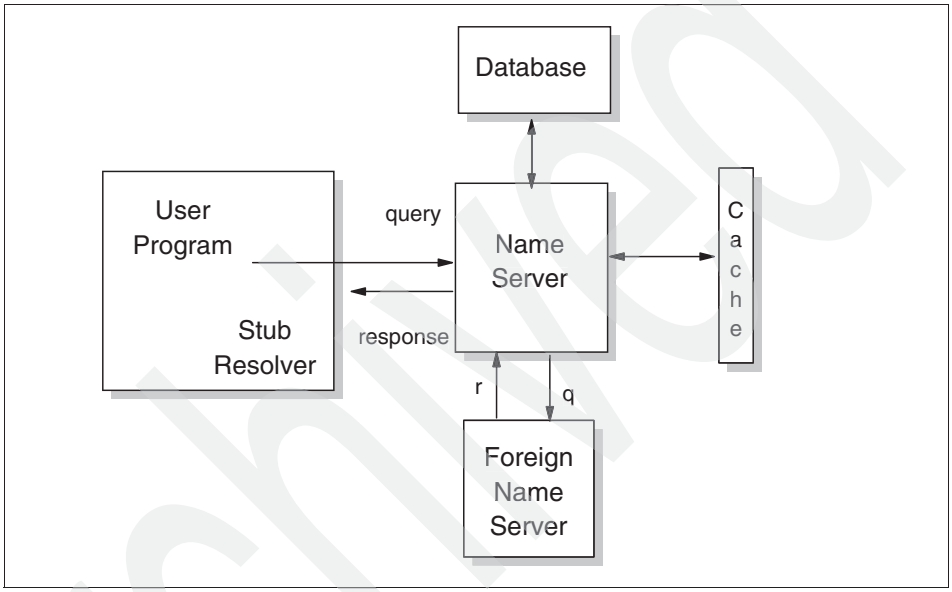


Figure 12-3 DNS: Using a stub resolver for domain name resolution

Domain name resolver operation

Domain name queries can be one of two types: *recursive* or *iterative* (also called *non-recursive*). A flag bit in the domain name query specifies whether the client desires a recursive query, and a flag bit in the response specifies whether the server supports recursive queries. The difference between a recursive and an iterative query arises when the server receives a request for which it cannot supply a complete answer by itself. A recursive query requests that the server issues a query itself to determine the requested information and returns the complete answer to the client. An iterative query means that the name server returns what information it has available and also a list of additional servers for the client to contact to complete the query.

Domain name responses can be one of two types: *authoritative* and *non-authoritative*. A flag bit in the response indicates which type a response is. When a name server receives a query for a domain in a zone over which it has

authority, it returns all of the requested information in a response with the authoritative answer flag set. When it receives a query for a domain over which it does not have authority, its actions depend on the setting of the recursion desired flag in the query:

- If the recursion desired flag is set and the server supports recursive queries, it will direct its query to another name server. This will either be a name server with authority for the domain given in the query, or it will be one of the root name servers. If the second server does not return an authoritative answer (for example, if it has delegated authority to another server), the process is repeated.
- When a server (or a full resolver program) receives a response, it will cache it to improve the performance of repeat queries. The cache entry is stored for a maximum length of time specified by the originator in a 32-bit *time-to-live (TTL)* field contained in the response. A typical TTL value is 86,400 seconds (one day).
- If the recursion desired flag is not set or the server does not support recursive queries, it will return whatever information it has in its cache and also a list of additional name servers to be contacted for authoritative information.

Domain name server operation

Each name server has *authority* for zero or more zones. There are three types of name servers:

Primary A primary name server loads a zone's information from disk and has authority over the zone.

Secondary A secondary name server has authority for a zone, but obtains its zone information from a primary server using a process called *zone transfer*. To remain synchronized, the secondary name servers query the primary on a regular basis (typically three hours) and re-execute the zone transfer if the primary has been updated. A name server can operate as a primary or a secondary name server for multiple domains, or a primary for some domains and as a secondary for others. A primary or secondary name server performs all of the functions of a caching-only name server.

Caching-only A name server that does not have authority for any zone is called a caching-only name server. A caching-only name server obtains all of its data from primary or secondary name servers as required. It requires at least one NS record to point to a name server from which it can initially obtain information.

Archive

When a domain is registered with the root and a separate zone of authority established, the following rules apply:

- The domain must be registered with the root administrator.
- There must be an identified administrator for the domain.
- There must be at least two name servers with authority for the zone that are accessible from outside and inside the domain to ensure no single point of failure.

We also recommend that name servers that delegate authority apply these rules, because the delegating name servers are responsible for the behavior of name servers under their authority.

12.1.9 Domain Name System resource records

The Domain Name System's distributed database is composed of *resource records* (RRs), which are divided into classes for different kinds of networks. We only discuss the Internet class of records. Resource records provide a mapping between domain names and *network objects*. The most common network objects are the addresses of Internet hosts, but the Domain Name System is designed to accommodate a wide range of different objects.

A zone consists of a group of resource records, beginning with a Start of Authority (SOA) record. The SOA record identifies the domain name of the zone. There will be a name server (NS) record for the primary name server for this zone. There might also be NS records for the secondary name servers. The NS records are used to identify which of the name servers are authoritative (see “Domain name resolver operation” on page 434). Following these records are the resource records, which might map names to IP addresses or aliases to names.

The following figure shows the general format of a resource record (Figure 12-4).

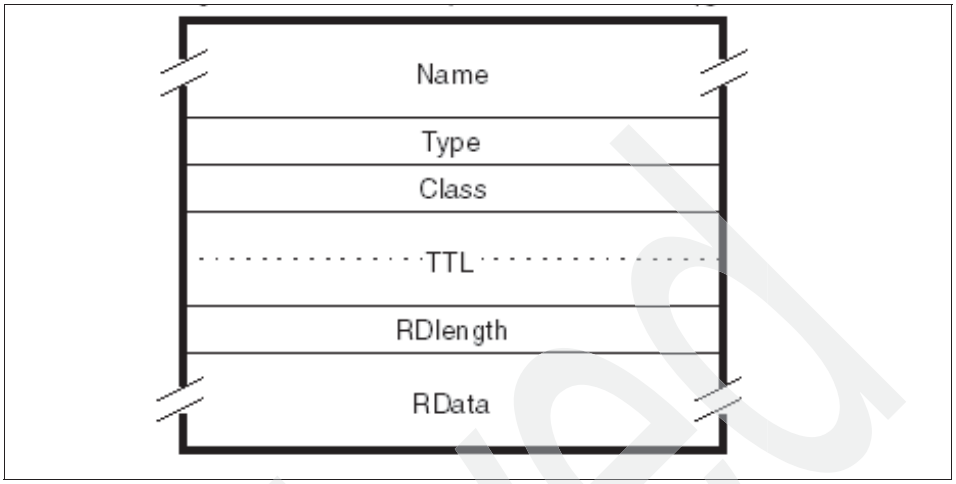


Figure 12-4 DNS general resource record format

Where:

Name The domain name to be defined. The Domain Name System is very general in its rules for the composition of domain names. However, it recommends a syntax for domain names that minimizes the likelihood of applications that use a DNS resolver (that is, nearly all TCP/IP applications) from misinterpreting a domain name. A domain name adhering to this recommended syntax will consist of a series of labels consisting of alphanumeric characters or hyphens, each label having a length of between 1 and 63 characters, starting with an alphabetic character. Each pair of labels is separated by a dot (period) in human-readable form, but not in the form used within DNS messages. Domain names are not case-sensitive.

Type Identifies the type of the resource in this record. There are numerous possible values, but some of the more common ones, along with the RFCs which define them, are listed in Table 12-2 on page 438.

Class Identifies the protocol family. The only commonly used value is IN (the Internet system), though other values are defined by RFC 1035 and include:

- CS (value 2): The CSNET class. This has been obsoleted.
- CH (value 3): The CHAOS class.
- HS (value 4): The Hesiod class.

TTL The time-to-live (TTL) time in seconds for which this resource record will be valid in a name server cache. This is stored in the DNS as an unsigned 32-bit value. A typical value for records pointing to IP addresses is 86400 (one day).

RDlength An unsigned 16-bit integer that specifies the length, in octets, of the RData field.

RData A variable length string of octets that describes the resource. The format of this information varies according to the Type and Class of the resource record.

Table 12-2 Some of the possible resource record types

Type	Value	Meaning	RFC def
A	1	A host address	1035
NS	2	An authoritative name server	1035
CNAME	5	The canonical name for an alias	1035
SOA	6	Marks the start of a zone of authority	1035
MB	7	A mailbox domain name (experimental)	1035
MG	8	A mail group member (experimental)	1035
MR	9	A mail rename domain name (experimental)	1035
NULL	10	A NULL resource record (experimental)	1035
WKS	11	A well-known service description	1035
PTR	12	A domain name pointer	1035
HINFO	13	Host information	1035
MINFO	14	Mailbox or mail list information	1035
MX	15	Mail exchange[a]	1035
TXT	16	Text strings	1035
RP	17	Responsible person record	1183
AFSDB	18	Andrew File System database	1183
X25	19	X.25 resource record	1183
ISDN	20	ISDN resource record	1183
RT	21	Route Through resource record	1183
NSAP	22	Network Service Access Protocol record	1348

Type	Value	Meaning	RFC def
NSAP-PTR	23	NSAP Pointer record	1348
KEY	25	The public key associated with a DNS name	2535
AAAA	28	An IPv6 address record	3596
LOC	29	GPS resource record	1876
SRV	33	Defines the services available in a zone	2872
CERT	37	Certificate resource records	4398
A6	38	Forward mapping of an IPv6 address	2874
DNAME	39	Delegation of IPv6 reverse addresses	2672
DS	39	Delegated Signer record (DNS security)	4034
RRSIG	46	Resource record digital signature	4034
NSEC	47	Next Secure record (DNS security)	4034
DNSKEY	48	Public Key record (DNS security	4034

a. The MX Type obsoletes Types MD (value 3, Mail destination) and MF (value 4, Mail forwarder).

12.1.10 Domain Name System messages

All messages in the Domain Name System protocol use a single format. This format is shown in Figure 12-5 on page 440. This frame is sent by the resolver to the name server. Only the header and the question section are used to form the query. Replies and forwarding of the query use the same frame, but with more sections filled in (the answer/authority/additional sections).

Archive

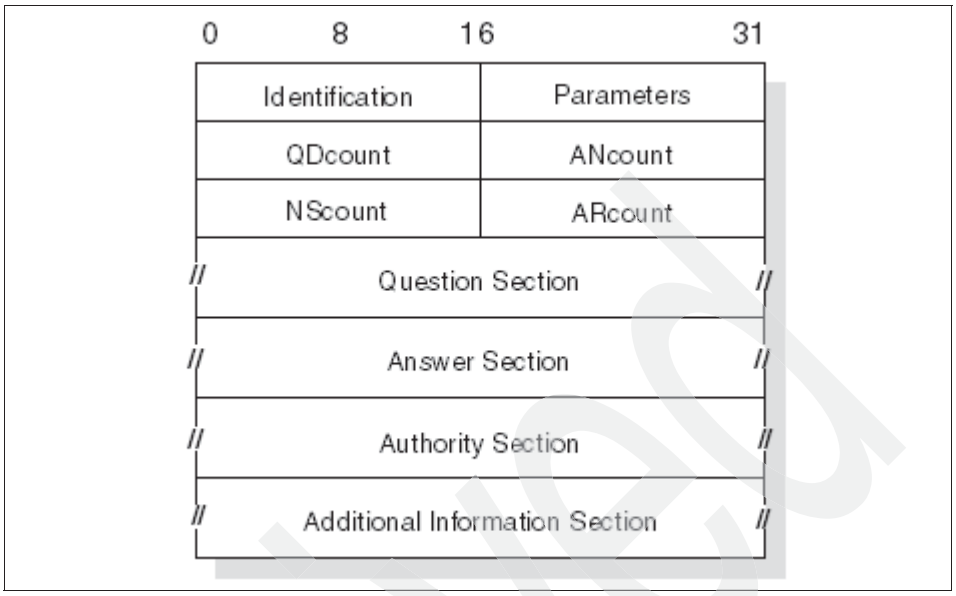


Figure 12-5 DNS message format

Header format

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length.

ID A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation of responses when multiple queries are outstanding at the same time.

Parameters A 16-bit value in the following format (Table 12-3).

Table 12-3 Parameters

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QR	Op code				AA	TC	RD	RA	Zero			Rcode			

Where:

QR Flag identifying a query (0) or a response(1).

Op code 4-bit field specifying the kind of query:

0 Standard query (QUERY)
1 Inverse query (IQUERY)
2 Server status request (STATUS)

Archived

Other values are reserved for future use:

AA Authoritative answer flag. If set in a response, this flag specifies that the responding name server is an authority for the domain name sent in the query.

TC Truncation flag. Set if message was longer than permitted on the physical channel.

RD Recursion desired flag. This bit signals to the name server that recursive resolution is asked for. The bit is copied to the response.

RA Recursion available flag. Indicates whether the name server supports recursive resolution.

Zero 3 bits reserved for future use. Must be zero.

Rcode 4-bit response code. Possible values are:

- **0** No error.
- **1** Format error. The server was unable to interpret the message.
- **2** Server failure. The message was not processed because of a problem with the server.
- **3** Name error. The domain name in the query does not exist. This is only valid if the AA bit is set in the response.
- **4** Not implemented. The requested type of query is not implemented by name server.
- **5** Refused. The server refuses to respond for policy reasons. Other values are reserved for future use.

QDcount An unsigned 16-bit integer specifying the number of entries in the question section.

ANcount An unsigned 16-bit integer specifying the number of RRs in the answer section.

NScount An unsigned 16-bit integer specifying the number of name server RRs in the authority section.

ARcount An unsigned 16-bit integer specifying the number of RRs in the additional records section.

Archived

Question section

The next section contains the queries for the name server. It contains QDcount (usually 1) entries, each in the format shown in Figure 12-6.

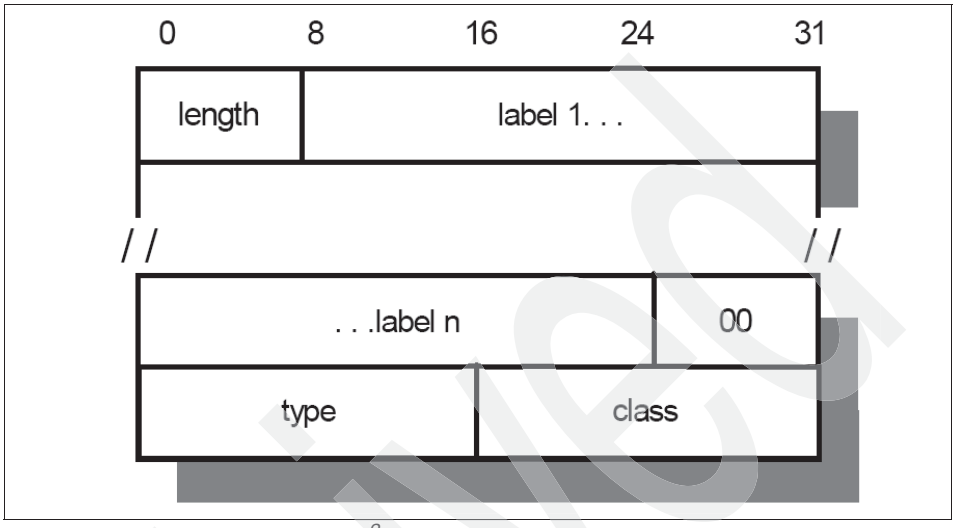


Figure 12-6 DNS question format[2]

Where:

Length A single byte giving the length of the next label.

Label One element of the domain name characters (for example, ibm from ral.ibm.com). The domain name referred to by the question is stored as a series of these variable length labels, each preceded by a 1-byte length.

00 X'00' indicates the end of the domain name and represents the null label of the root domain.

Type 2 bytes specifying the type of query. It can have any value from the Type field in a resource record.

Class 2 bytes specifying the class of the query. For Internet queries, this will be IN.

[2] Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format.

For example, the domain name mydiv.mycorp.com is encoded with the following fields:

```
X'05'
"mydiv"
X'06'
"mycorp"
X'03'
"com"
X'00'
```

Therefore, the entry in the question section for mydiv.mycorp.com requires 22 bytes: 18 to store the domain name and 2 each for the Qtype and Qclass fields.

Answer, authority, and additional resource sections

These three sections contain a variable number of resource records. The number is specified in the corresponding field of the header. The resource records are in the format shown in Figure 12-7.

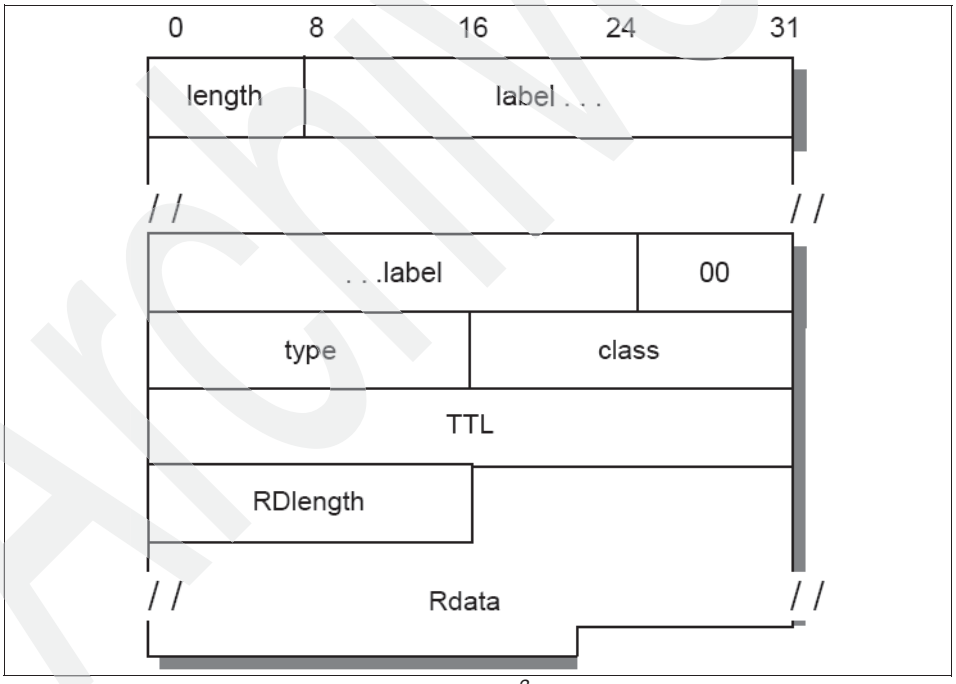


Figure 12-7 DNS: Answer Record Entry format [3]

[3] Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format.

Where the fields before the TTL field have the same meanings as for a question entry and:

TTL A 32-bit time-to-live value in seconds for the record. This defines how long it can be regarded as valid.

RDlength A 16-bit length for the Rdata field.

Rdata A variable length string whose interpretation depends on the Type field.

Message compression

In order to reduce the message size, a compression scheme is used to eliminate the repetition of domain names in the various RRs. Any duplicate domain name or list of labels is replaced with a pointer to the previous occurrence. The pointer has the form of a 2-byte field as shown in Figure 12-8.



Figure 12-8 DNS message compression

Where:

- The first 2 bits distinguish the pointer from a normal label, which is restricted to a 63-byte length plus the length byte ahead of it (which has a value of <64).
- The offset field specifies an offset from the start of the message. A zero offset specifies the first byte of the ID field in the header.
- If compression is used in an Rdata field of an answer, authority, or additional section of the message, the preceding RDlength field contains the real length after compression is done.

Refer to 12.2, “Dynamic Domain Name System” on page 453 for additional message formats.

Using the DNS Uniform Resource Identifiers (URI)

A DNS can also be queried using a Uniform Resource Identifier This is defined in RFC 4501. Strings are not case sensitive, and adhere to the following format:

```
“dns:” + [ “//” + dnsauthority + “:” + port + ”/” ] + dnsname +
[ “?” + dnsquery ]
```

Where:

dnsauthority The DNS server to which the query should be sent. If this is left blank, the query is sent to the default DNS server.

dnsname The name or IP address to be queried.

dnsquery The type of the query to be performed. This can be any combination, separated by a semicolon (;), of:

- **CLASS** Usually IN for internet, the class of the query
- **TYPE** The type of resource record desired

For example, a request using the URI to resolve www.myCorp.com to an IP address might appear as follows:

```
dns:www.mycorp.com
```

Additionally, the same request can be sent to the server at 10.1.2.3 on port 5353 using the following:

```
dns://10.1.2.3:5353/www.mycorp.com
```

Finally, this same query can be made specifying a CLASS of IN and a TYPE of A:

```
dns://10.1.2.3:5353/www.mycorp.com?class=IN;type=A
```

12.1.11 A simple scenario

Consider a stand-alone network (no outside connections), consisting of two physical networks:

- One has an Internet network address of 129.112.
- One has a network address of 194.33.7.

Archived

They are interconnected by an IP gateway (VM2). See Figure 12-9 for more details.

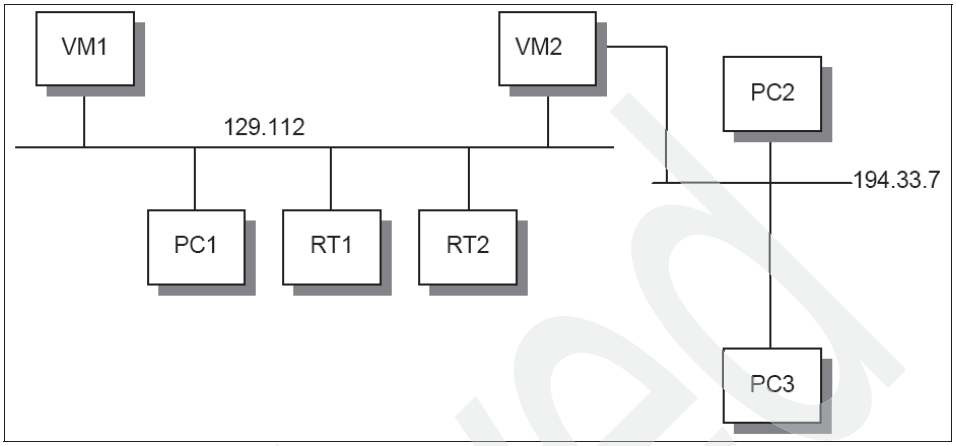


Figure 12-9 DNS: A simple configuration: Two networks connected by an IP gateway

Assume the name server function has been assigned to VM1. Remember that the domain hierarchical tree forms a logical tree, completely independent of the physical configuration. In this simple scenario, there is only one level in the domain tree, which will be referred to as *test.example*.

Archived

The zone data for the name server appears as shown in Figure 12-10 and continued in Figure 12-11 on page 448.

```
;note: an SOA record has no TTL field
;
$origin test.example.                                          ;note 1
@             IN SOA VM1.test.example. ADM.VM1.test.example.
                    (870611         ;serial number for data
                     1800           ;secondary refreshes every 30 mn
                     300            ;secondary reties every 5 mn
                     604800         ;data expire after 1 week
                     86400)         ;minimum TTL for data is 1 week
;
@       99999 IN NS  VM1.test.example.                         ;note 2
;
VM1     99999 IN A   129.112.1.1                               ;note 3
        99999 IN WKS 129.112.1.1 TCP (SMTP                     ;note 4
                                      FTP
                                      TELNET
                                      NAMESRV)
;
RT1     99999 IN A     129.112.1.2
              IN HINFO IBM RT/PC-AIX                           ;note 5
RT2     99999 IN A     129.112.1.3
              IN HINFO IBM RT/PC-AIX
PC1     99999 IN A     129.112.1.11
PC2     99999 IN A     194.33.7.2
PC3     99999 IN A     194.33.7.3
;
;VM2 is an IP gateway and has 2 different IP addresses
;
VM2     99999 IN A     129.112.1.4
        99999 IN A     194.33.7.1
        99999 IN WKS   129.112.1.4 TCP (SMTP FTP)
              IN HINFO IBM-3090-VM/CMS
```

Figure 12-10 Zone data for the name server, continued in Figure 12-11 on page 448

Archive

```
;
4.1.112.129.in-addr.arpa.  IN  PTR  VM2                        ;note 6
;
;Some mailboxes
;
central 10    IN MX  VM2.test.example.                  ;notes 7 and 8
;
;a second definition for the same mailbox, in case VM2 is down
;
central 20    IN MX  VM1.test.example.
waste   10    IN MX  VM2.test.example.
```

Figure 12-11 Zone data for the name server, continued from Figure 12-10 on page 447

Notes for Figure 12-10 on page 447 and Figure 12-11:

1. The $origin statement sets the @ variable to the zone name (test.example.). Domain names that do not end with a period are suffixed with the zone name. Fully qualified domain names (those ending with a period) are unaffected by the zone name.
2. Defines the name server for this zone.
3. Defines the Internet address of the name server for this zone.
4. Specifies well-known services for this host. These are expected to always be available.
5. Gives information about the host.
6. Used for inverse mapping queries (see 12.1.6, “Mapping IP addresses to domain names: Pointer queries” on page 430).
7. Will allow mail to be addressed to user@central.test.example.
8. See 15.1.2, “SMTP and the Domain Name System” on page 565 for the use of these definitions.

12.1.12 Extended scenario

Consider the case where a connection is made to a third network (129.113), which has an existing name server with authority for that zone (see Figure 12-12).

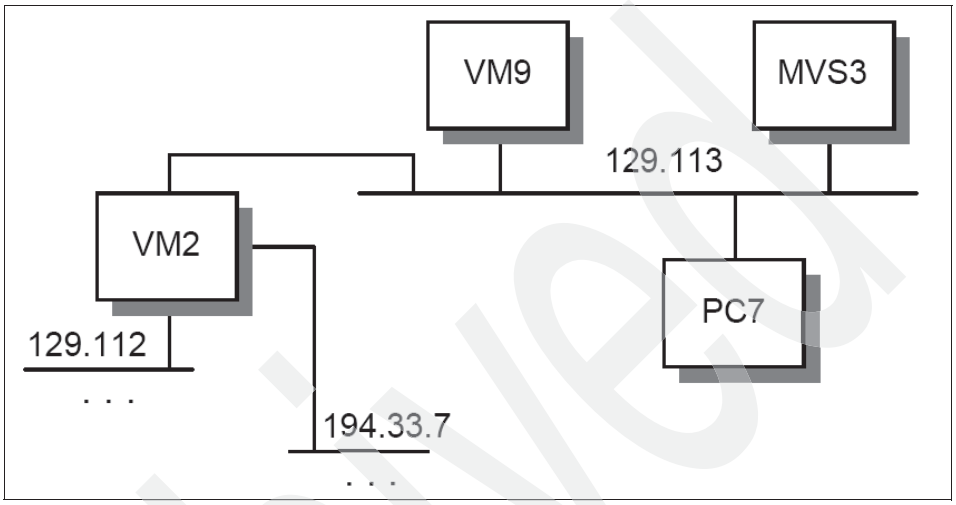


Figure 12-12 DNS: Extended configuration - Third network connected to existing configuration

Let us suppose that the domain name of the other network is tt.ibm.com and that its name server is located in VM9. All we have to do is add the address of this name server to our own name server database (in the named.ca initial cache file) and reference the other network by its own name server. The following two lines are all that is needed to do that:

```
tt.ibm.com.          99999  IN NS   VM9.tt.ibm.com.
VM9.tt.ibm.com.      99999  IN A    129.13.1.9
```

This simply indicates that VM9 is the authority for the new network and that all queries for that network will be directed to that name server.

12.1.13 Transport

Domain Name System messages are transmitted either as datagrams (UDP) or through stream connection (TCP):

- UDP usage: Server port 53 (decimal).

  Messages carried by UDP are restricted to 512 bytes. Longer messages are truncated and the truncation (TC) bit is set in the header. Because UDP frames can be lost, a retransmission strategy is required.

- TCP usage: Server port 53 (decimal).

  In this case, the message is preceded by a 2-byte field indicating the total message frame length.

- STD 3 – Host Requirements requires that:
  - A Domain Name System resolver or server that is sending a non-zone-transfer query *must* send a UDP query first. If the answer section of the response is truncated and if the requester supports TCP, it tries the query again using TCP. UDP is preferred over TCP for queries because UDP queries have much lower overall processing cost, and the use of UDP is essential for a heavily loaded server. Truncation of messages is rarely a problem given the current contents of the Domain Name System database, because typically 15 response records can be accommodated in the datagram, but this might change as new record types continue to be added to the Domain Name System.
  - TCP must be used for zone transfer activities because the 512-byte limit for a UDP datagram will always be inadequate for a zone transfer.
  - Name servers must support both types of transport.

As IPv6 becomes more pervasive throughout the Internet community, some problems are forecasted as a result of mixed IPv4/IPv6 network segments. Primarily, if a resolver that can only use IPv4 is forwarded to across a network segment that supports only IPv6, the resolver and the name server will be unable to communicate. As a result, the hierarchical namespace becomes fragmented into two sets of segments: those that support IPv4 and IPv6, and those that support only IPv6. This impending issue has been named the Problem of Name Space Fragmentation, and documented in RFC 3901. In order to preserve namespace continuity, RFC 3901 recommends the following:

- Every recursive name server should be either IPv4 only, or dual IPv4 and IPv6.
- Every DNS zone should be served by at least one IPv4-reachable authoritative name server.

Additional suggestions about configuring IPv6 DNS servers are in RFC 4339.

Dynamic DNS (DDNS)

The Dynamic Domain Name System (DDNS) is a protocol that defines extensions to the Domain Name System to enable DNS servers to accept requests to add, update, and delete entries in the DNS database dynamically. Because DDNS offers a functional superset to existing DNS servers, a DDNS server can serve both static and dynamic domains at the same time, a welcome feature for migration and overall DNS design.

DDNS is currently available in a non-secure and a secure flavor, defined in RFC 2136 and RFC 3007, respectively. Rather than allowing any host to update its DNS records, the secure version of DDNS uses public key security and digital signatures to authenticate update requests from DDNS hosts.

Without client authentication, another host could impersonate an unsuspecting host by remapping the address entry for the unsuspecting host to that of its own. After the remapping occurs, important data, such as logon passwords and mail intended for the host would, unfortunately, be sent to the impersonating host instead.

See 12.2, “Dynamic Domain Name System” on page 453 for more information about how DDNS works together with DHCP to perform seamless updates of reverse DNS mapping entries, and see 9.4, “DNS in IPv6” on page 367 for more information about DNS with IPv6.

12.1.14 DNS applications

Three common utilities for querying name servers are provided with many DNS implementations: **host**, **nslookup**, and **dig**. Though specifics details about each utility vary slightly by platform, most platforms provide a common set of options.

host

The **host** command obtains an IP address associated with a host name, or a host name associated with an IP address. The typical syntax for **host** command is:

```
host [options] name [server]
```

Where:

options Valid options typically include:

- **-c class** The query class. By default, this is IN (Internet), but other valid class names include CS (CSNET), CH (CHAOS), HS (Hesiod), and ANY (a wildcard encompassing all four classes).
- **-r** Disables recursive processing.

-t type The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).

-w Instructs the **host** command to wait forever for a reply.

name The name of the host or the address to be resolved.

server The name server to query.

nslookup

The **nslookup** command enables you to locate information about network nodes, examine the contents of a name server database, and establish the accessibility of name servers. The typical syntax for the **nslookup** command is:

```
nslookup [options] [host] [-nameserver]
```

Where:

options These options vary widely by platform. Refer to the documentation for a specific implementation for information about what options are available.

host The host name or IP address to be located.

-nameserver The name server to which the query is to be directed.

dig

dig stands for Domain Internet Groper, and enables you to exercise name servers, gather large volumes of domain name information, and execute simple domain name queries. The typical syntax for the **dig** command is:

```
dig @server [options] [name] [type] [class] [queryopt]
```

Where:

@server The DNS name server to be queried.

options Valid options typically include:

-b address The source IP address of the query-to address.

-c class The query class. By default, this is IN (Internet), but other valid class names include CS (CSNET), CH (CHAOS), and HS (Hesiod).

-f filename Causes **dig** to operate in batch mode, and specifies the file from which the batch commands can be found.

-p port	Specifies that **dig** should send the query to a port other than well-known DNS port 53.
-x address	Instructs **dig** to do a reverse lookup on the specified address.
name	The name of the resource record to be looked up.
type	The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).

12.2 Dynamic Domain Name System

The Domain Name System described in 12.1, “Domain Name System (DNS)” on page 426 is a static implementation without recommendations with regard to security. In order to implement DNS dynamically, take advantage of DHCP, and still to be able to locate any specific host by means of a meaningful label (such as its host name), the following extensions to DNS are required:

- A method for the host name to address a mapping entry for a client in the domain name server to be updated after the client has obtained an address from a DHCP server
- A method for the reverse address to host name mapping to take place after the client obtains its address
- Updates to the DNS to take effect immediately, without the need for intervention by an administrator
- Updates to the DNS to be authenticated to prevent unauthorized hosts from accessing the network and to stop imposters from using an existing host name and remapping the address entry for the unsuspecting host to that of its own
- A method for primary and secondary DNS servers to quickly forward and receive changes as entries are being updated dynamically by clients

However, implementation of a Dynamic Domain Name System (DDNS) can introduce problems if the environment is not secure. One method of security employed by DNS is the use of Secret Key Transaction Authentication (TSIG), defined in RFC 2845. This can be used to authenticate dynamic updates from clients, or authenticate responses coming from a recursive server. Additionally, these messages can now be protected for integrity and confidentiality through using TSIG over the Generic Security Service (GSS-TSIG). This extension, and the associated algorithms needed to implement GSS-TSIG, are defined in RFC 3645.

In addition to TSIG, and GSS-TSIG, several RFCs extended the functionality of DNS such that it incorporated additional security methods. These additions, defined in RFC 4033 and referred to as the DNS Security Extensions (DNSSEC), allow DNS to authenticate the origin of data as well as negative responses to DNS queries. However, they do not provide confidentiality, access control lists, or protection against denial-of-service-attacks. New resource records relating to security were added by RFCs 4034 and 4398, and include:

- DNSKEY (public key)
- DS (delegation signer)
- RRSIG (resource record digital signature)
- NSEC (authenticated denial of existence)
- CERT (public key certificates)

Note that these RRs are also listed in Table 12-2 on page 438. Specific details about how the DNS protocol was modified to take advantage of these additions is in RFC 4035.

12.2.1 Dynamic updates in the DDNS

The DNS message format (shown in Figure 12-5 on page 440) was designed for the querying of a static DNS database. RFC 2136 defines a modified DNS message for updates, called the UPDATE DNS message, illustrated in Figure 12-13 on page 455. This message adds or deletes resource records in the DNS, and allows updates to take effect without the DNS having to be reloaded.

Archived

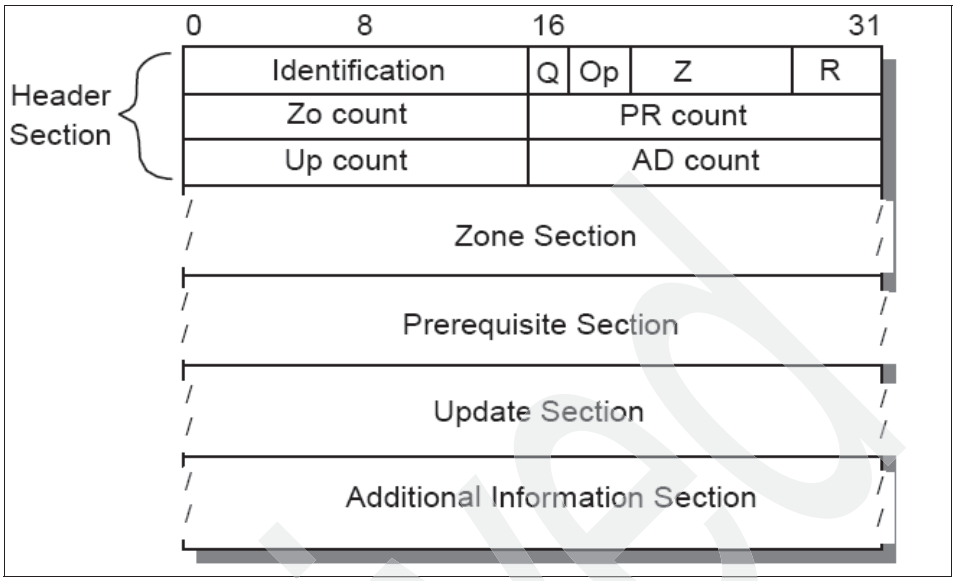


Figure 12-13 DDNS UPDATE message format

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length. They are:

Identification A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation when multiple queries/updates are outstanding at the same time.

Q Flag identifying an update request (0) or a response (1).

Op Opcode. The value 5 indicates an UPDATE message.

z 7-bit field set to 0 and reserved for future use.

R Response code (undefined in update requests). Possible values are:

- **0** No error.
- **1** Format error. The server was unable to interpret the message.
- **2** Server failure. The message was not processed due to a problem with the server.
- **3** Name error. A name specified does not exist.
- **4** Not implemented. The type of message specified in Opcode is not supported by this server.
- **5** Refused. The server refuses to perform the UPDATE requested for security or policy reasons.

Archived

6 Name error. A name exists when it should not.
7 RRset error. A resource record set exists when it should not.
8 RRset error. A resource record set specified does not exist.
9 Zone Authority error. The server is not authoritative for the zone specified.
10 Zone error. A name specified in the Prerequisite or Update sections is not in the zone specified.

ZO count The number of RRs in the Zone section.

PR count The number of RRs in the Prerequisite section.

UP count The number of RRs in the Update section.

AD count The number of RRs in the Additional information section.

Zone section This section is used to indicate the zone of the records that are to be updated. As all records to be updated must belong to the same zone, the zone section has a single entry specifying the zone name, zone type (which must be SOA), and zone class.

Prerequisite section
This section contains RRs or RRsets that either must, or must not, exist, depending on the type of update.

Update section This section contains the RRs, RRsets, or both that are to be added to or deleted from the zone.

Additional information section
This section can be used to pass additional RRs that relate to the update operation in process.

For further information about the UPDATE message format, refer to RFC 2136.

12.2.2 Incremental zone transfers in DDNS

RFC 1995 introduces the IXFR DNS message type, which allows incremental transfers of DNS zone data between primary and secondary DNS servers. In other words, when an update has been made to the zone data, only the change has to be copied to the other DNS servers that maintain a copy of the zone data, rather than the whole DNS database (as is the case with the AXFR DNS message type).

The format of an IXFR query is exactly that of a normal DNS query, but with a query type of IXFR. The response's answer section, however, is made up of *difference sequences*.

Each list difference sequences is preceded by the server's current version of the SOA and represents one update to the zone. Similarly, each difference sequence is preceded by an SOA version (indicating in which versions correspond to each change, and the difference sequences are ordered oldest to newest. Upon receiving this message, a server can update its zone by tracking the version history listed in the IXFR answer section.

For example, assume a server has the following zone:

```
MYZONE.MYDIV.MYCORP         IN SOA MYHOST.MYDIV.MYCORP (
                                   1 600 600 3600000 614800)
                            IN NS MYHOST.MYDIV.MYCORP
MYHOST.MYDIV.MYCORP         IN A  10.1.2.3
OTHERHOST.MYDIV.MYCORP      IN A  10.2.3.4
```

Otherhost.mydiv.mycorp is removed, and in version 2, thishost.mydiv.mycorp is added, leaving the zone as:

```
MYZONE.MYDIV.MYCORP         IN SOA MYHOST.MYDIV.MYCORP (
                                   2 600 600 3600000 614800)
                            IN NS MYHOST.MYDIV.MYCORP
MYHOST.MYDIV.MYCORP         IN A  10.1.2.3
THISHOST.MYDIV.MYCORP       IN A  10.2.3.5
```

If the server receives an IXFR query, it sends back the following answer section:

```
MYZONE.MYDIV.MYCORP         IN SOA serial=2
MYZONE.MYDIV.MYCORP         IN SOA serial=1
OTHERHOST.MYDIV.MYCORP      IN A  10.1.2.4
MYZONE.MYDIV.MYCORP         IN SOA serial=2
THISHOST.MYDIV.MYCORP       IN A  10.2.3.5
MYZONE.MYDIV.MYCORP         IN SOA serial=2
```

Note: If a server received an IXFR query, but incremental zone transfers are not available, it will send back the entire zone in the reply.

12.2.3 Prompt notification of zone transfer

RFC 1996 introduces the NOTIFY DNS message type, which is used by a master server to inform subordinate servers that an update has taken place and that they should initiate a query to discover the new data. The NOTIFY message uses the DNS message format, but only a subset of the available fields (unused fields are filled with binary zeros). The message is similar to a QUERY message, and can contain the name of the RRs that have been updated. Upon receipt of a NOTIFY message, the subordinate returns a response. The response contains no useful information, and only serves to alert the master server of receipt of the

NOTIFY. Based on the RRs contained in the notify, subordinate servers might then send an update query to the server to obtain the new changes.

12.3 Network Information System (NIS)

The Network Information System (NIS) is not an Internet standard. It was developed by Sun Microsystems, Inc. It was originally known as the Yellow Pages (YP) and many implementations continue to use this name.

NIS is a distributed database system that allows the sharing of system information in UNIX-based environment. Examples of system information that can be shared include the /etc/passwd, /etc/group, and /etc/hosts files. NIS has the following advantages:

- Provides a consistent user ID and group ID name space across a large number of systems
- Reduces the time and effort by users in managing their user IDs, group IDs, and NFS file system ownerships
- Reduces the time and effort by system administrators in managing user IDs, group IDs, and NFS ownerships

NIS is built on the RPC, and employs the client/server model. Most NIS implementations use UDP. However, because it uses RPC, it is also possible for it to be implemented over TCP. A NIS domain is a collection of systems consisting of:

NIS master server — Maintains *maps*, or databases, containing the system information, such as passwords and host names. These are also referred to as Database Maps (DBMs).

NIS subordinate server(s) — Can be defined to offload the processing from the master NIS server or when the NIS master server is unavailable.

NIS client(s) — The remaining systems that are served by the NIS servers.

The NIS clients do not maintain NIS maps; they query NIS servers for system information. Any changes to an NIS map is done only to the NIS master server (through RPC). The master server then propagates the changes to the NIS subordinate servers.

Note that the speed of a network determines the performance and availability of the NIS maps. When using NIS, the number of subordinate servers should be tuned in order to achieve these goals.

Because NIS is not standardized by the IETF, implementations vary by platform. However, most platforms make available the following common NIS commands:

makedbm	Generate a DBM file from an input file.
ypcat	Display the contents of a DBM file.
ypinit	Set up an NIS master or subordinate server.
ypmake	Performs the same function as **makedbm**, but provides the option to push the resulting DBMs to subordinate servers.
ypmatch	Prints the values associated with one or more keys in a DBM.
yppasswd	Change a login password stored in a DBM.
yppush	Pushes DBMs to subordinate servers.
ypwhich	Indicates what NIS server a client is using.
ypxfr	Pulls a DBM from the master server.

12.4 Lightweight Directory Access Protocol (LDAP)

When implementing a Distributed Computing Environment (DCE), directory services are automatically included because they are an intrinsic part of the DCE architecture. However, though widely used, implementation of a DCE is not a practical solution for every company needing directory services because it is an “all-or-nothing” architecture. As such, if the other services provided by a DCE are not required, or if implementation of the DCE model is not feasible (for example, if it is not feasible to install the client software on every workstation within the network), other directory service alternatives must be identified.

One such alternative is the Lightweight Directory Access Protocol (LDAP), which is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory, and is gaining wide acceptance as the directory access method of the Internet. It is supported by a growing number of software vendors and is being incorporated into a growing number of applications.

For further information about LDAP, refer to the IBM Redbook *Understanding LDAP - Design and Implementation*, SG24-4986.

12.4.1 LDAP: Lightweight access to X.500

The OSI directory standard, X.500, specifies that communication between the directory client and the directory server uses the Directory Access Protocol (DAP). However, as an application layer protocol, DAP requires the entire OSI protocol stack to operate, which requires more resources than are available in many small environments. Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol was desired.

LDAP was developed as a *lightweight* alternative to DAP, because it requires the more popular TCP/IP protocol stack rather than the OSI protocol stack. LDAP also simplifies some X.500 operations and omits some esoteric features. Two precursors to LDAP appeared as RFCs issued by the IETF, RFC 1202 – Directory Assistance Service and RFC 1249 – DIXIE Protocol Specification. These were both informational RFCs which were not proposed as standards. The directory assistance service (DAS) defined a method by which a directory client communicates to a proxy on an OSI-capable host, which issues X.500 requests on the client's behalf. DIXIE is similar to DAS, but provides a more direct translation of the DAP.

The first version of LDAP was defined in RFC 1487 – X.500 Lightweight Access, which was replaced by RFC 1777 – Lightweight Directory Access Protocol. Much of the work on DIXIE and LDAP was carried out at the University of Michigan, which provides reference implementations of LDAP and maintains LDAP-related Web pages and mailing lists. Since then, LDAPv2 has been replaced by LDAP Version 3. LDAPv3 is summarized in RFC 4510, but the technical specifications are divided into multiple subsequent RFCs listed in Table 12-4.

Table 12-4 LDAP-related RFCs

RFC number	Content
4510	Technical Specification Road Map
4511	The Protocol
4512	Directory Information Models
4513	Authentication Methods and Security Mechanisms
4514	String Representation of Distinguished Names
4515	String Representation of Search Filters
4516	Uniform Resource Locator
4517	Syntaxes and Matching Rules
4518	Internationalized String Preparation

RFC number	Content
4519	Schema for User Applications
4520	Internet Assigned Numbers Authority (IANA) Considerations for LDAP
4521	Considerations for LDAP
4522	The Binary Encoding Option
4523	Schema Definitions for X.509 Certificates
4524	COSINE/ LDAP X.500 Schema
4525	Modify-Increment Extension
4526	Absolute True and False Filters
4527	Read Entry Controls
4528	Assertion Control
4529	Requesting Attributes by Object Class in LDAP
4530	entryUUID Operational Attribute
4531	Turn Operation
4532	“Who Am I” Operation
4533	Content Synchronization Operation

Though an application program interface (API) for previous versions of LDAP was limited to specifications in RFC 1823, the LDAPv3 provides both a C API and a Java™ Naming and Directory Interface (JNDI).

12.4.2 The LDAP directory server

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. An application client program initiates an LDAP message by calling an LDAP API. But an X.500 directory server does not understand LDAP messages. In fact, the LDAP client and X.500 server even use different communication protocols (TCP/IP versus OSI). The LDAP client actually communicates with a gateway process (also called a proxy or front end) that forwards requests to the X.500 directory server (see Figure 12-14 on page 462), known as an LDAP server, which fulfils requests from the LDAP client. It does this by becoming a client of the X.500 server. The

LDAP server must communicate using both TCP/IP (with the client) and OSI (with the X.500 server).

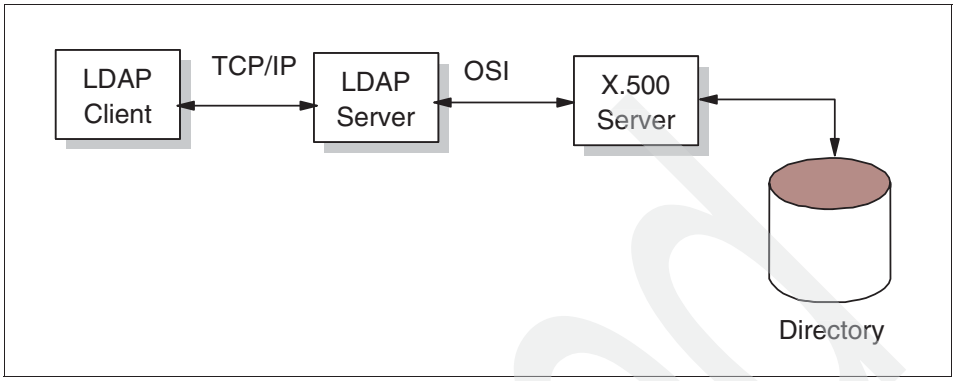


Figure 12-14 LDAP server acting as a gateway to an X.500 directory server

As the use of LDAP grew and its benefits became apparent, people who did not have X.500 servers or the environments to support them wanted to build directories that could be accessed by LDAP clients. This requires that the LDAP server store and access the directory itself instead of only acting as a gateway to X.500 servers (see Figure 12-15). This eliminates any need for the OSI protocol stack but, of course, makes the LDAP server much more complicated, because it must store and retrieve directory entries. These LDAP servers are often called stand-alone LDAP servers because they do not depend on an X.500 directory server. Because LDAP does not support all X.500 capabilities, a stand-alone LDAP server only needs to support the capabilities required by LDAP.

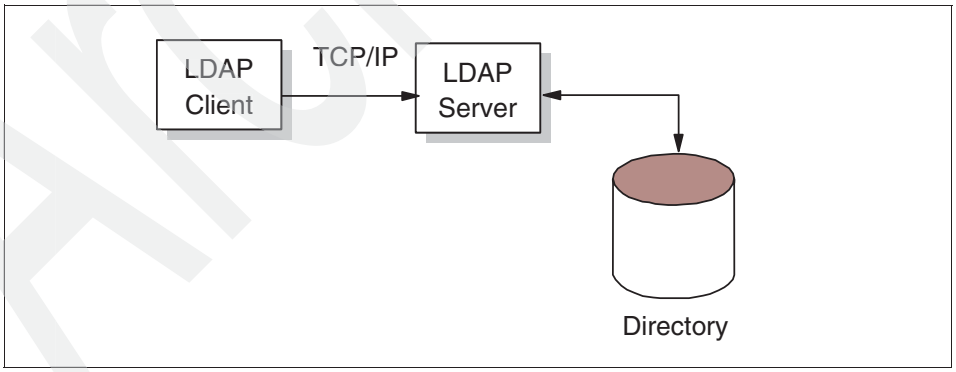


Figure 12-15 Stand-alone LDAP server

The concept of the LDAP server being able to provide access to local directories supporting the X.500 model, rather than acting only as a gateway to an X.500 server, is discussed in RFC 4511 (see Table 12-4 on page 460). From the client's

point of view, any server that implements the LDAP protocol is an LDAP directory server, whether the server actually implements the directory or is a gateway to an X.500 server. The directory that is accessed can be called an LDAP directory, whether the directory is implemented by a stand-alone LDAP server or by an X.500 server.

12.4.3 Overview of LDAP architecture

LDAP defines the content and format of messages exchanged between an LDAP client and an LDAP server. The messages specify the operations requested by the client (search, modify, delete, and so on), the responses from the server, and the format of data carried in the messages. LDAP messages are carried over TCP/IP, a connection-oriented protocol, so there are also operations to establish and disconnect a session between the client and server.

The general interaction between an LDAP client and an LDAP server takes the following form:

1. The client establishes a session with an LDAP server. This is known as *binding* to the server. The client specifies the host name or IP address and TCP/IP port number where the LDAP server is listening. The client can provide a user name and a password to properly authenticate with the server, or the client can establish an anonymous session with default access rights. The client and server can also establish a session that uses stronger security methods, such as encryption of data (see 12.4.5, “LDAP security” on page 471).
2. The client then performs operations on directory data. LDAP offers both read and update capabilities. This allows directory information to be managed as well as queried. LDAP supports searching the directory for data meeting arbitrary user-specified criteria. Searching is the most common operation in LDAP. A user can specify what part of the directory to search and what information to return. A search filter that uses Boolean conditions specifies which directory data matches the search.
3. When the client has finished making requests, it closes the session with the server. This is also known as *unbinding*.

Because LDAP was originally intended as a lightweight alternative to DAP for accessing X.500 directories, the LDAP server follows an X.500 model. The directory stores and organizes data structures known as entries. A directory entry usually describes an object such as a person, a printer, a server, and so on. Each entry has a name called a distinguished name (DN) that uniquely identifies it. The DN consists of a sequence of parts called relative distinguished names (RDNs), much like a file name can consist of a path of directory names. The entries can be arranged into a hierarchical tree-like structure based on their

distinguished names. This tree of directory entries is called the directory information tree (DIT).

LDAP defines operations for accessing and modifying directory entries, such as:

- Searching for entries meeting user-specified criteria
- Adding an entry
- Deleting an entry
- Modifying an entry
- Modifying the distinguished name or relative distinguished name of an entry (move)
- Comparing an entry

12.4.4 LDAP models

LDAP can be better understood by considering the four models upon which it is based:

Information Describes the structure of information stored in an LDAP directory.

Naming Describes how information in an LDAP directory is organized and identified.

Functional Describes the operations that can be performed on the information stored in an LDAP directory.

Security Describes how the information in an LDAP directory can be protected from unauthorized access.

The following sections discuss the first three LDAP models. We describe LDAP security in 12.4.5, “LDAP security” on page 471.

The information model

The basic unit of information stored in the directory is an entry, which represents an object of interest in the real world such as a person, server, or organization. Each entry contains one or more attributes that describe the entry. Each attribute has a type and one or more values. For example, the directory entry for a person might have an attribute called telephoneNumber. The syntax of the telephoneNumber attribute specifies that a telephone number must be a string of numbers that can contain spaces and hyphens. The value of the attribute is the person's telephone number, such as 123-456-7890 (a person might have multiple telephone numbers, in which case this attribute would have multiple values).

Archived

In addition to defining what data can be stored as the value of an attribute, an attribute syntax also defines how those values behave during searches and other directory operations. This is done using syntax and matching rules. The attribute telephoneNumber, for example, might have a syntax that specifies:

- Lexicographic ordering.
- Case, spaces, and dashes are ignored during the comparisons.
- Values must be character strings.

For example, using the correct definitions, the telephone numbers 123-456-7890, 123456-7890, and 1234567890 are considered to be the same. A few of the common syntaxes and matching rules, defined in RFC 4517, are listed in Table 12-5.

Table 12-5 Examples of LDAP syntaxes

Syntaxes and matching rules	Description
Bit String	A sequence of binary digits
Postal Address	A sequence of strings that form an address in a physical mail system
caseExactMatch	A matching rule requiring that string comparisons are case-sensitive
caseIgnoreMatch	A matching rule that does not require case-sensitive comparisons

Table 12-6 lists some common attributes defined by RFC 4519. Some attributes have alias names that can be used wherever the full attribute name is used.

Table 12-6 Examples of LDAP syntaxes

Attribute, alias	Syntax	Description	Example
commonName, cn	cis	Common name of an entry	John Smith
surname, sn	cis	A person's last name	Smith
initials	cis	A person's initials	JS
telephoneNumber	tel	A person's telephone number	123-456-7890

An object class is a general description, sometimes called a template, of an object type, as opposed to the description of a specific object of that type. For example, the object class *person* has a surname attribute, while the object describing John Smith has a surname attribute with the value *Smith*. The object

classes that a directory server can store and the attributes they contain are described by a *schema*. A schema defines where object classes are allowed in the directory, which attributes they must contain, which attributes are optional, and the syntax of each attribute. For example, a schema might define a *person* object class that requires that a person has a character-string surname attribute, can optionally have a number-string telephoneNumber attribute, and so on. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schemas also define the inheritance and subclassing of objects, and where in the DIT structure (hierarchy) objects can appear.

Though an implementation can define any schema to meet its needs, RFC 4519 defines a few standard schemas. Table 12-7 lists a few of the common schema (object classes and their required attributes). In many cases, an entry can consist of more than one object class.

Table 12-7 Examples of object classes and required attributes

Object class	Description	Attributes
country	Defines a country	Required: countryCode Optional: searchGuide description
locality	Defines a place in the physical world	Required: None Optional: street seeAlso searchGuide description
person	Defines a person	Required: surname commonName Optional: userPassword telephoneNumber seeAlso description

Because servers can define their own schema, LDAP includes the functionality of allowing a client to query a server for the contents of the supported schema.

The naming model

The LDAP naming model defines how entries are identified and organized. Entries are organized in a tree-like structure called the directory information tree (DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DNs are made up of a sequence of relative distinguished names (RDNs). Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry.

Each RDN is derived from the attributes of the directory entry. In the simple and common case, an RDN has the form <attribute-name>=<value>. A DN is composed of a sequence of RDNs separated by commas. These relationships are defined in RFC 4514.

An example of a DIT is shown in Figure 12-16. The example is very simple, but can be used to illustrate some basic concepts. Each box represents a directory entry. The root directory entry is conceptual and does not actually exist. Attributes are listed inside each entry. The list of attributes shown is not complete. For example, the entry for the country UK (`c=UK`) could have an attribute called `description` with the value `United Kingdom`.

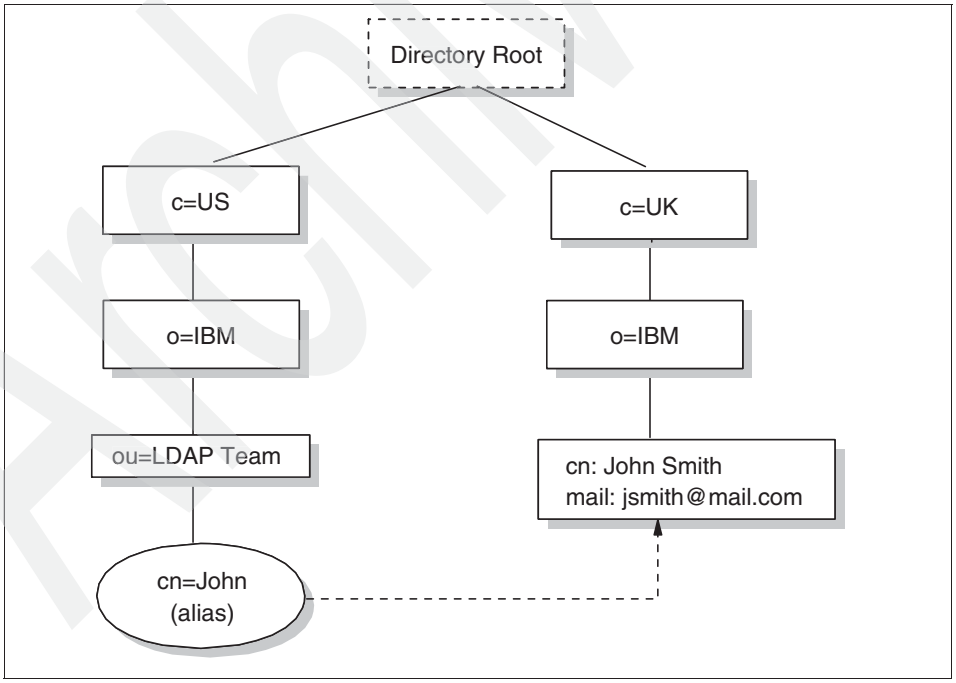


Figure 12-16 Example of a directory information tree (DIT)

It is usual to follow either a geographical or an organizational scheme to position entries in the DIT. For example, entries that represent countries would be at the top of the DIT. Below the countries would be national organizations, states, and provinces, and so on. Below this level, entries might represent people within those organizations or further subdivisions of the organization. The lowest layers of the DIT entries can represent any object, such as people, printers, application servers, and so on. The depth or breadth of the DIT is not restricted and can be designed to suit application requirements.

Entries are named according to their position in the DIT. The directory entry in the lower-right corner of Figure 12-16 on page 467 has the DN `cn=John Smith,o=IBM,c=UK`.

Note: DNs read from leaf-to-root, as opposed to names in a file system directory, which usually read from root-to-leaf.

The DN is made up of a sequence of RDNs. Each RDN is constructed from an attribute (or attributes) of the entry it names. For example, the DN `cn=John Smith,o=IBM,c=UK` is constructed by adding the RDN `cn=John Smith` to the DN of the ancestor entry `o=IBM,c=UK`.

The DIT is described as being tree-like, implying that it is not a tree. This is because of aliases. Aliases allow the tree structure to be circumvented. This can be useful if an entry belongs to more than one organization or if a commonly used DN is too complex. Another common use of aliases is when entries are moved within the DIT and you want access to continue to work as before. In Figure 12-16 on page 467, `cn=John,ou=LDAP Team,o=IBM,c=US` is an alias for `cn=John Smith,o=IBM,c=UK`.

Because an LDAP directory can be distributed, an individual LDAP server might not store the entire DIT. Instead, it might store the entries for a particular department but not the entries for the ancestors of the department. For example, a server might store the entries for the Accounting department at Yredbookscorp. The highest node in the DIT stored by the server would be `ou=Accounting,o=Yredbookscorp,c=US`. The server would store entries `ou=Accounting,o=Yredbookscorp,c=US` but not for `c=US` or for `o=Yredbookscorp,c=US`. The highest entry stored by a server is called a *suffix*. Each entry stored by the server ends with this suffix, so in this case, the suffix is the entire `ou=Accounting,o=Yredbookscorp,c=US`.

A single server can support multiple suffixes. For example, in addition to storing information about the Accounting department, the same server can store information about the Sales department at MyCorp. The server then has the suffixes `ou=Accounting,o=Yredbookscorp,c=US` and `ou=Sales,o=MyCorp,c=US`. Because a server might not store the entire DIT, servers need to be linked

together in some way in order to form a distributed directory that contains the entire DIT. This is accomplished with *referrals*. A referral acts as a pointer to an entry on another LDAP server where requested information is stored. A referral is an entry of objectClass *referral*. It has an attribute, *ref*, whose value is the LDAP URL of the referred entry on another LDAP server. See 12.4.6, “LDAP URLs” on page 474 for further information. Referrals allow a DIT to be partitioned and distributed across multiple servers. Portions of the DIT can also be replicated. This can improve performance and availability.

Note: When an application uses LDAP to request directory information from a server, but the server only has a referral for that information, the LDAP URL for that information is passed to the client. It is then the responsibility of that client to contact the new server to obtain the information. This is unlike the standard mechanisms of both DCE and X.500, where a directory server, if it does not contain the requested information locally, will always obtain the information from another server and pass it back to the client.

The functional model

LDAP defines operations for accessing and modifying directory entries. LDAP operations can be divided into the following three categories:

Query Includes the search and compare operations used to retrieve information from a directory.

Update Includes the add, delete, modify, modify RDN, and unsolicited notification operations used to update stored information in a directory. These operations will normally be carried out by an administrator.

Authentication Includes the bind, unbind, abandon, and startTLS operations used to connect and disconnect to and from an LDAP server, establish access rights, and protect information. For further information, see 12.4.5, “LDAP security” on page 471.

The search operation

The most common operation is the search. This operation is very flexible and therefore has some of the most complex options. The search operation allows a client to request that an LDAP server search through some portion of the DIT for information meeting user-specified criteria in order to read and list the results.

The search can be very general or very specific. The search operation allows the specification of the starting point within the DIT, how deep within the DIT to search, the attributes an entry must have to be considered a match, and the attributes to return for matched entries.

Some example searches expressed informally are:

- Find the postal address for `cn=John Smith,o=IBM,c=UK`.
- Find all the entries that are children of the entry `ou=ITSO,o=IBM,c=US`.
- Find the e-mail address and phone number of anyone in an organization whose last name contains the characters “miller” and who also has a fax number.

To perform a search, the following parameters must be specified:

Base A DN that defines the starting point, called the base object, of the search. The base object is a node within the DIT.

Scope Specifies how deep within the DIT to search from the base object. There are three choices:

- **baseObject** Only the base object is examined.
- **singleLevel** Only the immediate children of the base object are examined; the base object itself is not examined.
- **wholeSubtree** The base object and all of its descendants are examined.

Alias dereferencing Specifies if aliases are dereferenced. That is, the actual object of interest, pointed to by an alias entry, is examined. Not dereferencing aliases allows the alias entries themselves to be examined. This parameter must be one of the following:

- **neverDerefAliases** Do not deference aliases.
- **derefInSearching** Dereference aliases only when searching subordinates of the base object.
- **derefFindingBaseObj** Dereference aliases only when searching for the base object, but not when searching subordinates of the base object.
- **derefAlways** Always dereference aliases.

Size Limit The maximum number of entries that should be returned as a result of the search.

Time Limit The maximum number of seconds allowed to perform the search. Specifying zero indicates that there is no time limit.

Types Only This parameter has two possible values:

- **TRUE** Only attribute descriptions are returned.
- **FALSE** Attribute descriptions and values are returned.

Search filter Specifies the criteria an entry must match to be returned from a search. The search filter is a Boolean combination of attribute value assertions. An attribute value assertion tests the value of an attribute for equality, less than or equal, and so on.

Attributes to return Specifies which attributes to retrieve from entries that match the search criteria. Because an entry can have many attributes, this allows the user to only see the attributes in which they are interested.

12.4.5 LDAP security

Security is of great importance in the networked world of computers, and this is true for LDAP as well. When sending data over insecure networks, internally or externally, sensitive information might need to be protected during transportation. There is also a need to know who is requesting the information and who is sending it. This is especially important when it comes to the update operations on a directory. RFC 4513 discusses the authentication methods and security mechanisms available in LDAPv3, which can be divided into the following sections:

Authentication Assurance that the opposite party (machine or person) really is who he/she/it claims to be.

Integrity Assurance that the information that arrives is really the same as what was sent.

Confidentiality Protection against information disclosure, by means of data encryption, to those who are not intended to receive it.

Authorization Assurance that a party is really allowed to do what it is requesting to do, usually checked after user authentication. Authorization is achieved by assigning access controls, such as read, write, or delete, for user IDs or common names to the resources being accessed. Because these attributes are usually platform-specific, LDAP does not provide specific controls. Instead, it has

built-in mechanisms to allow the use of the platform-specific controls.

Because the use of authorization controls is platform-specific, the following sections describe only the authentication, integrity, and confidentiality. There are several mechanisms that can be used for this purpose; the most important ones are discussed here. These are:

- No authentication
- Basic authentication
- Simple Authentication and Security Layer (SASL)
- Transport Layer Security (TLS)

The security mechanism to be used in LDAP is negotiated when the connection between the client and the server is established.

No authentication

No authentication should only be used when data security is not an issue and when no special access control permissions are involved. This might be the case, for example, when your directory is an address book browsable by anybody. No authentication is assumed when you leave the password and DN field empty in the bind API call. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access (not to be confused with the SASL anonymous user discussed in “Simple Authentication and Security Layer (SASL)”).

Basic authentication

Basic authentication is also used in several other Web-related protocols, such as HTTP. When using basic authentication with LDAP, the client identifies itself to the server by means of a DN and a password, which are sent in the clear over the network (some implementations might use Base64 encoding instead). The server considers the client authenticated if the DN and password sent by the client matches the password for that DN stored in the directory. Base64 encoding is defined in the Multipurpose Internet Mail Extensions, or MIME (see 15.3, “Multipurpose Internet Mail Extensions (MIME)” on page 571). Base64 is a relatively simple encryption, and it is not hard to break after the data has been captured in the network.

Simple Authentication and Security Layer (SASL)

SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols, and is defined in RFC 4422. It has been added to LDAPv3 to overcome the authentication shortcomings of Version 2. SASL was originally devised to add stronger authentication to the IMAP protocol (see 15.5,

"Internet Message Access Protocol (IMAP4)" on page 591), but has since evolved into a more general system for mediating between protocols and authentication systems.

In SASL, connection protocols, such as LDAP, IMAP, and so on, are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. A complete list of SASL profiles can be obtained from the Information Sciences Institute (ISI). Among these are IMAP, SMTP, POP, and LDAP. Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. Optionally, a security layer can be negotiated to encrypt the data after authentication and ensure confidentiality. LDAPv3 includes such a command (**`ldap_sasl_bind()`** or **`ldap_sasl_bind_s()`**). The key parameters that influence the security method used are:

dn This is the distinguished name of the entry which is to bind. This can be thought of as the user ID in a normal user ID and password authentication.

mechanism This is the name of the security method to use. Valid security mechanisms are, currently:

- **OTP** The One Time Password mechanism (defined in RFC 2444).
- **GSSAPI** The Generic Security Services Application Program Interface (defined in RFC 2743).
- **CRAM-MD5** The Challenge/Response Authentication Mechanism (defined in RFC 2195), based on the HMAC-MD5 MAC algorithm.
- **DIGEST-MD5** An HTTP Digest-compatible CRAM based on the HMAC -MD5 MAC algorithm.
- **EXTERNAL** An external mechanism. Usually this is TLS, discussed in "Transport Layer Security (TLS)" on page 474.
- **ANONYMOUS** Unauthenticated access.

credentials This contains the arbitrary data that identifies the DN. The format and content of the parameter depends on the mechanism chosen. If it is, for example, the ANONYMOUS mechanism, it can be an arbitrary string or an e-mail address that identifies the user.

SASL provides a high-level framework that lets the involved parties decide on the particular security mechanism to use. The SASL security mechanism negotiation between client and server is done in the clear. After the client and the server

Archive

agree on a common mechanism, the connection is secure against modifying the authentication identities. However, an attacker might try to eavesdrop the mechanism negotiation and cause a party to use the least secure mechanism. In order to prevent this from happening, configure clients and servers to use a minimum security mechanism, provided they support such a configuration option. As stated earlier, SSL and its successor, TLS, are the mechanisms commonly used in SASL for LDAP. For details about these protocols, refer to 22.7, “Secure Sockets Layer (SSL)” on page 854.

Because no data encryption method was specified in LDAPv2, some vendors added their own SSL calls to the LDAP API. A potential drawback of such an approach is that the API calls might not be compatible among different vendor implementations. The use of SASL, as specified in LDAPv3, assures compatibility, although it is likely that vendor products will support only a subset of the possible range of mechanisms (possibly only SSL).

Transport Layer Security (TLS)

Transport Layer Security (TLS) is available through the SASL EXTERNAL method, described earlier. An LDAP client can opt to secure a session using TLS at any point during a transaction with an LDAP server, except when:

- The session is already TLS protected.
- A multi-stage SASL negotiation is in progress.
- The client is awaiting a response from an operation request.

To request that TLS be set up on the session, the client sends a *StartTLS* message to the server. This enables the client and server to exchange certificates. RFC 4513 requires that, in addition to this, the client verifies the server’s identity using the DNS or IP address presented in the server’s certificate. This prevents a client’s attempt to connect to a server from being intercepted by malicious user, who might then stage a *man-in-the-middle* attack. After this has occurred, the client and server can negotiate a ciphersuite.

12.4.6 LDAP URLs

Because LDAP has become an important protocol on the Internet, a URL format for LDAP resources has been defined in RFC 4516. LDAP URLs begin with ldap:// or ldaps://, if the LDAP server communicates using SSL. LDAP URLs can simply name an LDAP server, or can specify a complex directory search.

Some examples help make the format of LDAP URLs clear. The following example refers to the LDAP server on the host ldpserv.mydiv.mycorp.com (using the well-known port 389):

```
ldap://ldpserv.mydiv.mycorp.com/
```

Additionally, search options can be specified in the URL. The following example retrieves all the attributes for the DN `ou=Accounting,c=US` from the LDAP server on host ldpserv.mydiv.mycorp.com. In this case, nonstandard port 4389 is explicitly specified here as an example.

```
ldap://ldpserv.mydiv.mycorp.com:4389/ou=Accounting,c=US
```

The following example retrieves all the attributes for the DN `cn=JohnSmith,ou=Sales,o=myCorp,c=US`. Note that some characters are considered unsafe in URLs because they can be removed or treated as delimiters by some programs. Unsafe characters such as space, comma, brackets, and so forth need to be represented by their hexadecimal value preceded by the percent sign:

```
ldap://ldpserv.mydiv.mycorp.com/cn=John%20Smith,o=myCorp,c=US
```

In this example, `%20` is a space. More information about unsafe characters and URLs in general are in RFC 4516.

In addition to options, the URL can specify what values attributes are to be returned using the ? symbol. For example, assume we want to find the U.S. address of myCorp. We use the following URL:

```
ldap://ldpserv.mydiv.mycorp.com:4389/o=myCorp,c=US?postalAddress
```

12.4.7 LDAP and DCE

DCE has its own Cell Directory Service, or CDS (see 13.3.1, “DCE directory service” on page 498). If applications never access resources outside of their DCE cell, only CDS is required. However, if an application needs to communicate with resources in other DCE cells, the Global Directory Agent (GDA) is required. The GDA accesses a global (that is, non-CDS) directory where the names of DCE cells can be registered. This global directory (GDS) can be either a Domain Name System (DNS) directory or an X.500 directory. The GDA retrieves the address of a CDS server in the remote cell. The remote CDS can then be contacted to find DCE resources in that cell. Using the GDA enables an organization to link multiple DCE cells together using either a private directory on an intranet or a public directory on the Internet.

In view of LDAP's strong presence in the Internet, two LDAP projects have been sponsored by The Open Group to investigate LDAP integration with DCE technology.

LDAP interface for the GDA

One way LDAP is being integrated into DCE is to allow DCE cells to be registered in LDAP directories. The GDA in a cell that wants to connect to remote cells is configured to enable access to the LDAP directory (see Figure 12-17).

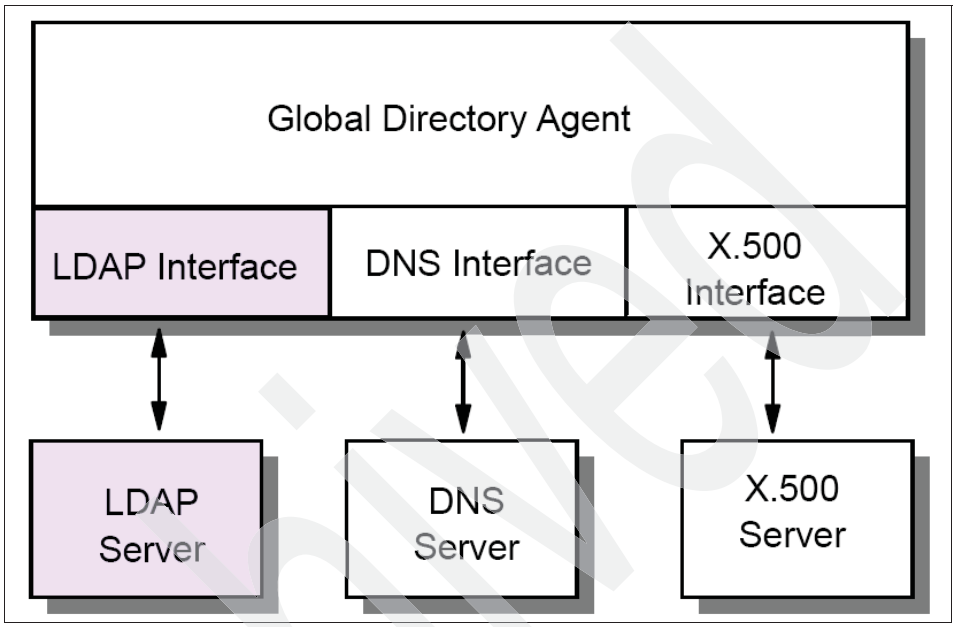


Figure 12-17 The LDAP interface for GDA

DCE originally only supported X.500 and DNS name syntax for cell names. LDAP and X.500 names both follow the same hierarchical naming model, but their syntax is slightly different. X.500 names are written in reverse order and use a slash (/) rather than a comma (,) to separated relative distinguished names. When the GDA is configured to use LDAP, it converts cell names in X.500 format into the LDAP format.

LDAP interface for the CDS

DCE provides two programming interfaces to the Directory Service; Name Service Interface (NSI) and the X/Open Directory Service (XDS). XDS is an X.500-compatible interface used to access information in the GDS, and it can also be used to access information in the CDS. However, the use of NSI is much more common in DCE applications. The NSI API provides functionality that is specifically tailored for use with DCE client and server programs that use RPC. NSI allows servers to register their address and the type of RPC interface they support. This address/interface information is called an RPC binding, and is

needed by clients that want to contact the server. NSI allows clients to search the CDS for RPC binding information.

NSI was designed to be independent of the directory where the RPC bindings are stored. However, the only supported directory to date has been CDS. NSI will be extended to also support adding and retrieving RPC bindings from an LDAP directory. This will allow servers to advertise their RPC binding information in either CDS or an LDAP directory. Application programs can use either the NSI or the LDAP API when an LDAP directory is used (see Figure 12-18).

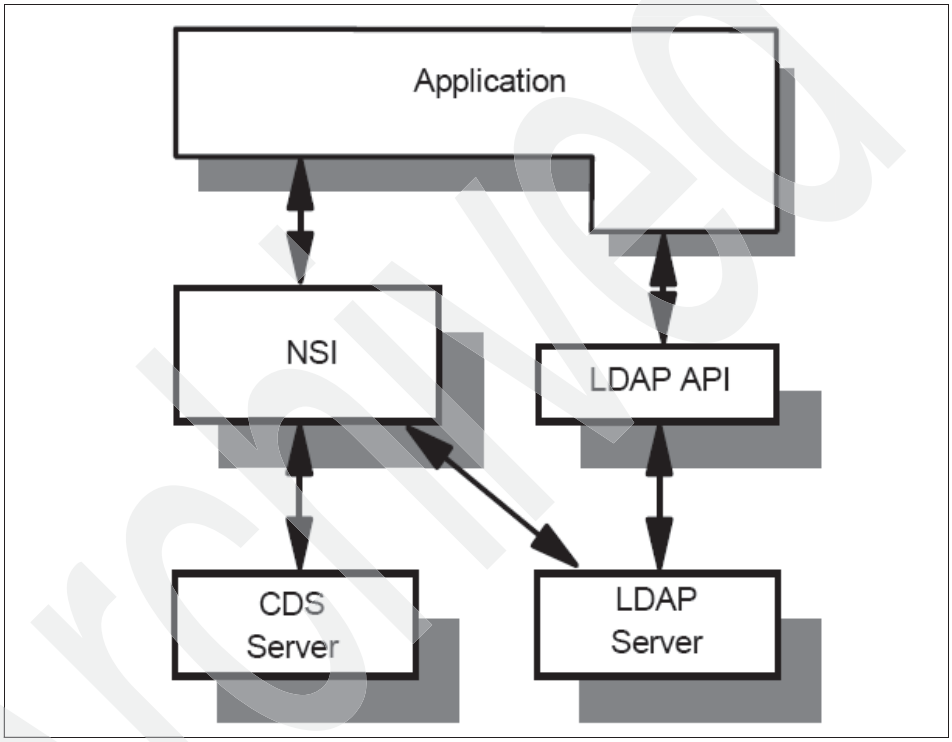


Figure 12-18 The LDAP interface for NSI

12.4.8 The Directory-Enabled Networks (DEN) initiative

In September 1997, Cisco Systems Inc. and Microsoft® Corp. announced the so-called Directory-Enabled Networks (DEN) initiative as a result of a collaborative work. Many companies, such as IBM, either support this initiative or actively participate in ad hoc working groups (ADWGs). DEN represents an information model specification for an integrated directory that stores information about people, network devices, and applications. The DEN schema defines the object classes and their related attributes for those objects. As such, DEN is a

key piece to building intelligent networks, where products from multiple vendors can store and retrieve topology and configuration-related data.

Of special interest is that the DEN specification defines LDAPv3 as the core protocol for accessing DEN information, which makes information available to LDAP-enabled clients or network devices, or both.

DEN makes use of the Common information Model (CIM). CIM details a way of integrating different management models such as SNMP MIBs and DMTF MIFs. At the time of writing, the most current CIM schema was version 2.12, released in April of 2006.

More information about the DEN initiative can be found on the founder's Web at:

http://www.dmtf.org/standards/wbem/den/
http://www.dmtf.org/standards/cim/

12.4.9 Web-Based Enterprise Management (WBEM)

WBEM is a set of standards designed to deliver an integrated set of management tools for the enterprise. By making use of XML and CIM, it becomes possible to manage network devices, desktop systems, telecom systems and application systems, all from a Web browser. For further information, see:

http://www.dmtf.org/standards/wbem/

12.5 RFCs relevant to this chapter

The following RFCs provide detailed information about the directory and naming protocols and architectures presented throughout this chapter:

- RFC 1032 – Domain administrators guide (November 1987)
- RFC 1033 – Domain administrators operations guide (November 1987)
- RFC 1034 – Domain names - concepts and facilities (November 1987)
- RFC 1035 – Domain names - implementation and specifications (November 1987)
- RFC 1101 – DNS encoding of network names and other types (April 1989)
- RFC 1183 – New DNS RR Definitions (October 1990)
- RFC 1202 – Directory Assistance service (February 1991)
- RFC 1249 – DIXIE Protocol Specification (August 1991)
- RFC 1348 – DNS NSAP RRs (July 1992)

- RFC 1480 – The US Domain (June 1993)
- RFC 1706 – DNS NSAP Resource Records (October 1994)
- RFC 1823 – The LDAP Application Programming Interface (August 1995)
- RFC 1876 – A Means for Expressing Location Information in the Domain Name System (January 1996)
- RFC 1995 – Incremental Zone Transfer in DNS (August 1996)
- RFC 1996 – A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY) (August 1996)
- RFC 2136 – Dynamic Updates in the Domain Name System (DNS UPDATE) (April 1997)
- RFC 2444 – The One-time-Password SASL Mechanism (October 1998)
- RFC 4592 – The Role of Wildcards in the Domain Name System (July 2006)
- RFC 2743 – Generic Security Service Application Program Interface Version 2, Update 1 (January 2000)
- RFC 2874 – DNS Extensions to Support IPv6 Address Aggregation and Renumbering (July 2000)
- RFC 3007 – Secure Domain Name Systems (DNS) Dynamic Update (November 2000)
- RFC 3494 – Lightweight Directory Access protocol version 2 (LDAPv2) (March 2003)
- RFC 3596 – DNS Extensions to Support IP Version 6 (October 2003)
- RFC 3645 – Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG) (October 2003)
- RFC 3901 – DNS IPv6 Transport Operational Guidelines (September 2004)
- RFC 4033 – DNS Security Introduction and Requirements (March 2005)
- RFC 4034 – Resource Records for the DNS Security Extensions (March 2005)
- RFC 4035 – Protocol Modifications for the DNS Security Extensions (March 2005)
- RFC 4339 – IPv6 Host Configuration of DNS Server Information Approaches (February 2006)
- RFC 4398 – Storing Certificates in the Domain Name System (DNS) (March 2006)
- RFC 4422 – Simple Authentication and Security Layer (SASL) (June 2006)
- RFC 4501 – Domain Name System Uniform Resource Identifiers (May 2006)

Archive

- RFC 4505 – Anonymous Simple Authentication and Security Layer (SASL) (June 2006)
- RFC 4510 – Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map (June 2006)
- RFC 4511 – Lightweight Directory Access Protocol (LDAP): The Protocol (June 2006)
- RFC 4512 – Lightweight Directory Access Protocol (LDAP): Directory Information Models (June 2006)
- RFC 4513 – Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms (June 2006)
- RFC 4514 – Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names (June 2006)
- RFC 4515 – Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters (June 2006)
- RFC 4516 – Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator (June 2006)
- RFC 4517 – Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules (June 2006)
- RFC 4518 – Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation (June 2006)
- RFC 4519 – Lightweight Directory Access Protocol (LDAP): Schema for User Applications (June 2006)
- RFC 4520 – Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) (June 2006)
- RFC 4521 – Considerations for Lightweight Directory Access Protocol (LDAP) (June 2006)
- RFC 4522 – Lightweight Directory Access Protocol (LDAP): The Binary Encoding Option (June 2006)
- RFC 4523 – Lightweight Directory Access Protocol (LDAP): Schema Definitions for X.509 Certificates (June 2006)
- RFC 4524 – Lightweight Directory Access Protocol (LDAP): COSINE/LDAP X.500 Schema (June 2006)
- RFC 4525 – Lightweight Directory Access Protocol (LDAP): Modify-Increment Extension (June 2006)
- RFC 4526 – Lightweight Directory Access Protocol (LDAP): Absolute True and False Filters (June 2006)

Archived

- RFC 4527 – Lightweight Directory Access Protocol (LDAP): Read Entry Controls (June 2006)
- RFC 4528 – Lightweight Directory Access Protocol (LDAP): Assertion Control (June 2006)
- RFC 4529 – Requesting Attributes by Object Class in the Lightweight Directory Access Protocol (LDAP) (June 2006)
- RFC 4530 – Lightweight Directory Access Protocol (LDAP): entryUUID Operational Attribute (June 2006)
- RFC 4531 – Lightweight Directory Access Protocol (LDAP): Turn Operation (June 2006)
- RFC 4532 – Lightweight Directory Access Protocol (LDAP): “Who Am I?” Operation (June 2006)
- RFC 4533 – Lightweight Directory Access Protocol (LDAP): Content Synchronization Operation (June 2006)

Archived

Archived

13

Remote execution and distributed computing

One of the most fundamental mechanisms employed on networked computers is the ability to execute on the remote systems. That is, a user wants to invoke an application on a remote machine. A number of application protocols exist to allow this remote execution capability, most notably the Telnet protocol. This chapter discusses some of these protocols. In addition, we discuss the concept of distributed computing.

© Copyright IBM Corp. 1989-2006. All rights reserved.

13.1 Telnet

Telnet is a standard protocol with STD number 8. Its status is recommended. It is described in RFC 854 – Telnet Protocol Specifications and RFC 855 – Telnet Option Specifications.

The Telnet protocol provides a standardized interface, through which a program on one host (the Telnet client) can access the resources of another host (the Telnet server) as though the client were a local terminal connected to the server. See Figure 13-1 for more details.

For example, a user on a workstation on a LAN can connect to a host attached to the LAN as though the workstation were a terminal attached directly to the host. Of course, Telnet can be used across WANs as well as LANs.

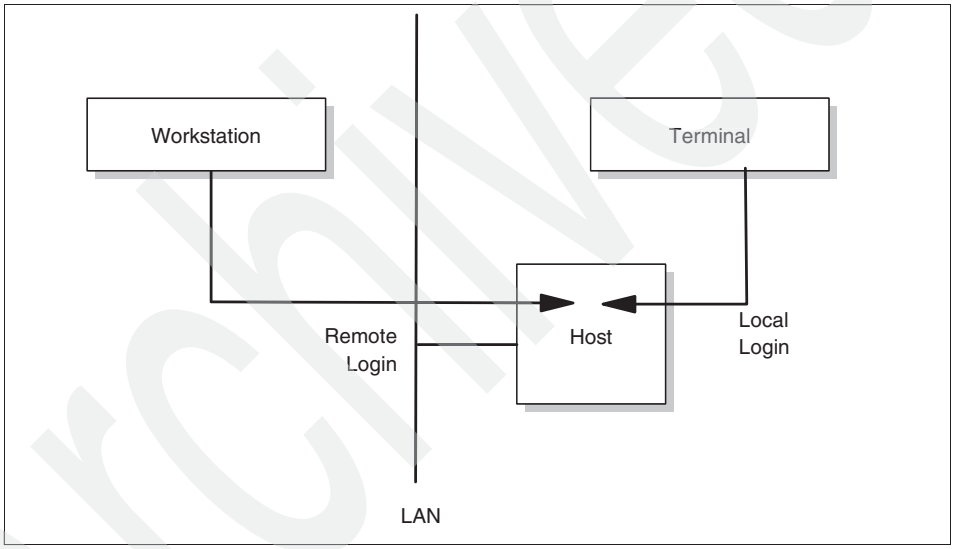


Figure 13-1 Telnet operation

Most Telnet implementations do not provide you with graphics capabilities.

13.1.1 Telnet operation

Telnet protocol is based on three ideas:

- The Network Virtual Terminal (NVT) concept. An NVT is an imaginary device with a basic structure common to a wide range of real terminals. Each host maps its own terminal characteristics to those of an NVT and assumes that every other host will do the same.

- A symmetric view of terminals and processes.
- Negotiation of terminal options. The principle of negotiated options is used by the Telnet protocol, because many hosts want to provide additional services, beyond those available with the NVT. Various options can be negotiated. The server and client use a set of conventions to establish the operational characteristics of their Telnet connection through the “DO, DONT, WILL, WONT” mechanism discussed later in this chapter.

The two hosts begin by verifying their mutual understanding. After this initial negotiation is complete, they are capable of working on the minimum level implemented by the NVT. After this minimum understanding is achieved, they can negotiate additional options to extend the capabilities of the NVT to reflect more accurately the capabilities of the real hardware in use. Because of the symmetric model used by Telnet (see Figure 13-2), both the host and the client can propose additional options to be used.

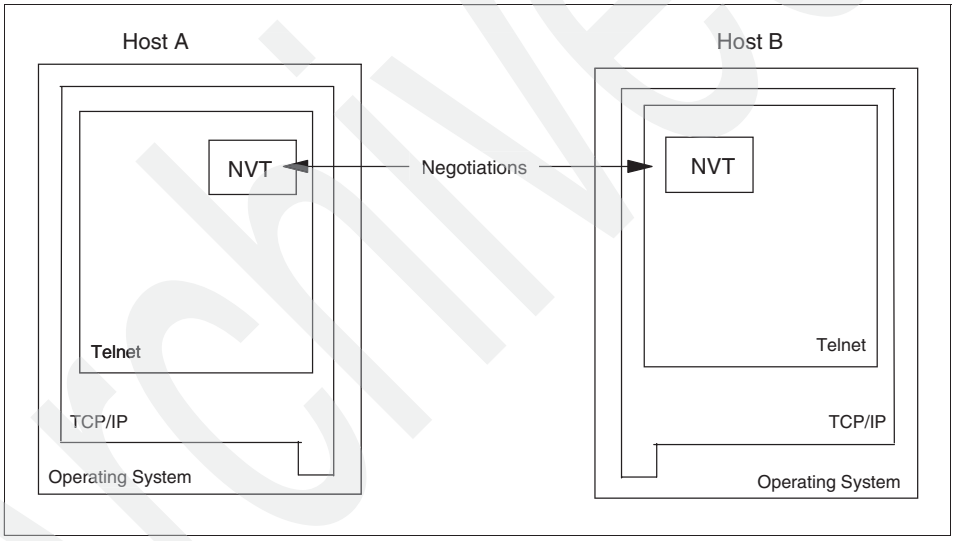


Figure 13-2 Telnet negotiations

13.1.2 Network Virtual Terminal

The NVT has a printer (or display) and a keyboard. The keyboard produces outgoing data, which is sent over the Telnet connection. The printer receives the incoming data. The basic characteristics of an NVT, unless they are modified by mutually agreed options, are:

- The data representation is 7-bit ASCII transmitted in 8-bit bytes.
- The NVT is a half-duplex device operating in a line-buffered mode.

- The NVT provides a local echo function.

All of these can be negotiated by the two hosts. For example, a local echo is preferred because of the lower network load and superior performance, but there is an option for using a remote echo (see Figure 13-3), although no host is required to use it.

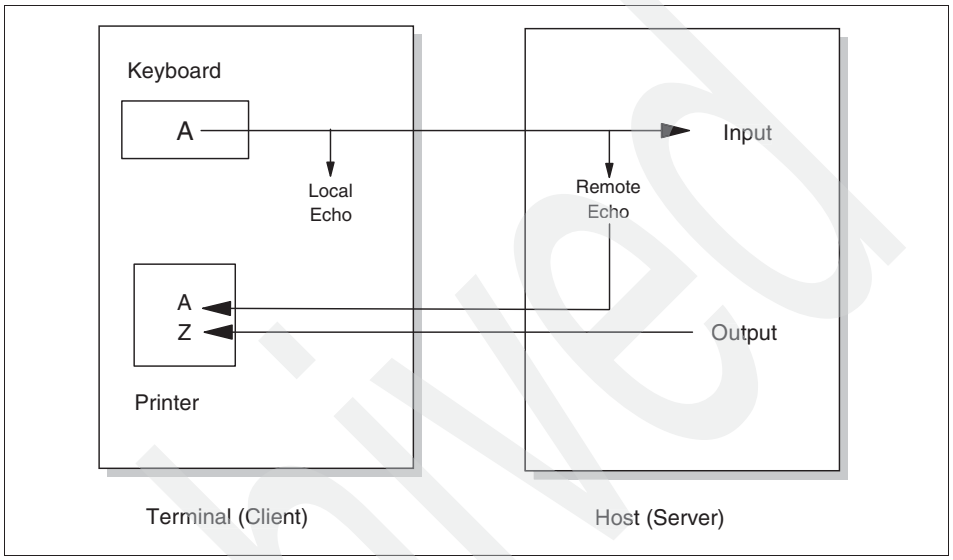


Figure 13-3 Telnet: Echo option

An NVT printer has an unspecified carriage width and page length. It can handle printable ASCII characters (ASCII code 32 to 126) and understands some ASCII control characters, such as those shown in Table 13-1.

Table 13-1 ASCII control characters

Command	ASCII	Action
NULL (NUL)	0	No operation.
Line feed (LF)	10	Moves printer to next line, keeping the same horizontal position.
Carriage return (CR)	13	Moves the printer to the next left margin.
BELL (BEL)	7	Produces and audible or visible signal.
Backspace (BS)	8	Moves print head one character position towards the left margin.

Archived

Command	ASCII	Action
Horizontal tab (HT)	9	Moves print head to the next horizontal tab stop.
Vertical tab (VT)	11	Moves printer to next vertical tab stop.
Form feed (FF)	12	Moves to top of next page, keeping the same horizontal position.

13.1.3 Telnet options

There is an extensive set of Telnet options, and the reader should consult STD 1 – Official Internet Protocol Standards for the standardization state and status for each of them. At the time of writing, the following options were defined (Table 13-2).

Table 13-2 Telnet options

Number	Name	State	RFC	STD
0	Binary transmission	Standard	856	27
1	Echo	Standard	857	28
3	Suppress Go Ahead	Standard	858	29
5	Status	Standard	859	30
6	Timing mark	Standard	860	31
255	Extended options list	Standard	861	32
34	Linemode	Draft	1184	
2	Reconnection	Proposed		
4	Approximate, message size negotiation	Proposed		
7	Remote controlled trans and echo	Proposed	726	
8	Output line width	Proposed		
9	Output page size	Proposed		
10	Output carriage-return disposition	Proposed	652	
11	Output horizontal tab stops	Proposed	653	

Number	Name	State	RFC	STD
12	Output horizontal tab disposition	Proposed	654	
13	Output form feed disposition	Proposed	655	
14	Output vertical tab stops	Proposed	656	
15	Output vertical tab disposition	Proposed	657	
16	Output line feed disposition	Proposed	658	
17	Extended ASCII	Proposed	698	
18	Logout	Proposed	727	
19	Byte macro	Proposed	735	
20	Data entry terminal	Proposed	1043	
21	SUPDUP	Proposed	736	
22	SUPDUP output	Proposed	749	
23	Send location	Proposed	779	
24	Terminal type	Proposed	1091	
25	End of record	Proposed	885	
26	TACACS user identification	Proposed	927	
27	Output marking	Proposed	933	
28	Terminal location number	Proposed	946	
29	Telnet 3270 regime	Proposed	1041	
30	X.3 PAD	Proposed	1053	
31	Negotiate window size	Proposed	1073	
32	Terminal speed	Proposed	1079	
33	Remote flow control	Proposed	1372	
35	X Display location	Proposed	1096	
39	Telnet environment option	Proposed	1572	
40	TN3270 enhancements	Proposed	1647	
37	Telnet authentication option	Experimental	1416	

Number	Name	State	RFC	STD
41	Telnet xauth	Experimental		
42	Telnet charset	Experimental	2066	
43	Telnet remote serial port	Experimental		
44	Telnet com port control	Experimental	2217	

All of the standard options have a status of recommended and the remainder have a status of elective. There is a historic version of the Telnet Environment option which is not recommended; it is Telnet option 36 and was defined in RFC 1408.

Full-screen capability

Full-screen Telnet is possible provided the client and server have compatible full-screen capabilities. For example, VM and MVS provide a TN3270-capable server. To use this facility, a Telnet client must support TN3270.

13.1.4 Telnet command structure

The communication between client and server is handled with internal commands, which are not accessible by users. All internal Telnet commands consist of 2- or 3-byte sequences, depending on the command type.

Archive

The Interpret As Command (IAC) character is followed by a command code. If this command deals with option negotiation, the command will have a third byte to show the code for the referenced option. See Figure 13-4 for more details.

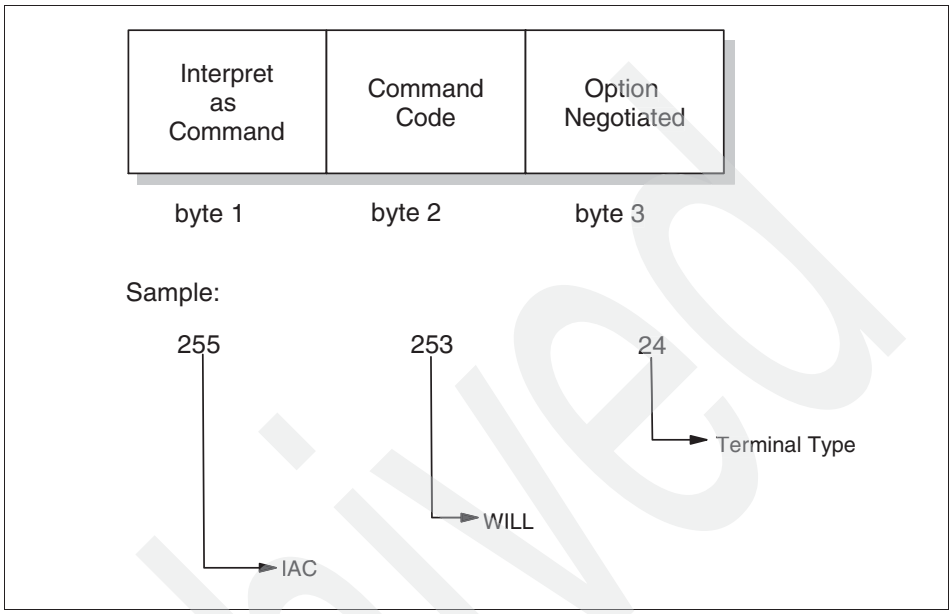


Figure 13-4 Telnet: Internal Telnet command proposes negotiation about terminal type

Table 13-3 shows some of the possible command codes.

Table 13-3 Command codes

Command	Code	Comments
SE	240	End of sub-negotiation parameters.
NOP	241	No operation.
Data Mark	242	The data stream portion of a sync. This must always be accompanied by a TCP urgent notification.
Break	243	NVT character BRK.
Go Ahead	249	The GA signal.
SB	250	Start of sub-negotiation of option indicated by the immediately following code.

Archive

Command	Code	Comments
WILL	251	Shows desire to use, or confirmation of using, the option indicated by the code immediately following.
WONT	252	Shows refusal to use or continue to use the option.
DO	253	Requests that other party uses, or confirms that you are expecting the other party to use, the option indicated by the code immediately following.
DONT	254	Demands that the other party stop using, or confirms that you are no longer expecting the other party to use, the option indicated by the code immediately following.
IAC	255	Interpret As command. Indicates that what follows is a Telnet command, not data.

13.1.5 Option negotiation

Using internal commands, Telnet is able to negotiate options in each host. The starting base of negotiation is the NVT capability: Each host to be connected must agree to this minimum. Every option can be negotiated by the use of the four command codes WILL, WONT, DO, and DONT. In addition, some options have suboptions: If both parties agree to the option, they use the SB and SE commands to manage the sub-negotiation. Table 13-4 shows a simplified example of how option negotiation works.

Table 13-4 Option negotiation

Send	Reply	Meaning
DO transmit binary	WILL transmit binary	
DO window size	WILL window size	Can we negotiate window size?
SB Window size 0 80 0 24 SE		Specify window size.
DO terminal type	WILL terminal type	Can we negotiate terminal type?
SB terminal type SE		Send me your terminal characteristics.

Send	Reply	Meaning
	SB terminal type IBM=3278-2 SE	My terminal is a 3278-2.
DO echo	WONT echo	

The terminal types are defined in STD 2 – Assigned Numbers.

13.1.6 Telnet basic commands

The primary goal of the Telnet protocol is the provision of a standard interface for hosts over a network. To allow the connection to start, the Telnet protocol defines a standard representation for some functions:

IP Interrupt Process

AO Abort Output

AYT Are You There

EC Erase Character

EL Erase Line

SYNCH Synchronize

13.1.7 Terminal emulation (Telnet 3270)

Telnet can be used to make a TCP/IP connection to an SNA host. However, Telnet 3270 is used to provide 3270 Telnet emulation (TN3270). The following differences between traditional Telnet and 3270 terminal emulation make it necessary for additional Telnet options specifically for TN3270 to be defined:

- 3270 terminal emulation uses block mode rather than line mode.
- 3270 terminal emulation uses the EBCDIC character set rather than the ASCII character set.
- 3270 terminal emulation uses special key functions, such as ATTN and SYSREQ.

The TN3270 connection over Telnet is accomplished by the negotiation of the following three different Telnet options:

- Terminal type
- Binary transmission
- End of record

A TN3270 server must support these characteristics during initial client/server session negotiations. Binary transmission and end of record options can be sent in any order during the TN3270 negotiation. Note that TN3270 does not use any additional options during the TN3270 negotiation; it uses normal Telnet options. After a TN3270 connection is established, additional options can be used. These options are TELNET-REGIME, SUPPRESS-GO-AHEAD, ECHO, and TIMING-MARK.

Terminal type option is a string that specifies the terminal type for the host, such as IBM 3278-3. The -3 following 3278 indicates the use of an alternate screen size other than the standard size of 24x80. The binary transmission Telnet option states that the connection will be other than the initial NVT mode. If the client or server want to switch to NVT mode, they send a command that disables the binary option. A 3270 data stream consists of a command and related data. Because the length of the data associated with the command can vary, every command and its related data must be separated with the IAC EOR sequence. For this purpose, the EOR Telnet option is used during the negotiation.

Other important issues for a TN3270 connection are the correct handling of the ATTN and SYSREQ functions. The 3270 ATTN key is used in SNA environments to interrupt the current process. The 3270 SYSREQ key is used in SNA environments to terminate the session without closing the connection. However, SYSREQ and ATTN commands cannot be sent directly to the TN3270 server over a Telnet connection. Most of the TN3270 server implementations convert the BREAK command to an ATTN request to the host through the SNA network. On the client side, a key or combination of keys are mapped to BREAK for this purpose. For the SYSREQ key, either a Telnet Interrupt Process command can be sent or a SYSREQ command can be sent imbedded into a TN3270 data stream. Similarly, on the client side, a key or combination of keys are mapped for SYSREQ.

There are some functions that cannot be handled by traditional TN3270. Some of these issues include:

- TN3270 does not support 328x types of printers.
- TN3270 cannot handle SNA BIND information.
- There is no support for the SNA positive/negative response process.
- TN3270 cannot map Telnet sessions into SNA device names.

13.1.8 TN3270 enhancements (TN3270E)

The 3270 structured field allows non-3270 data to be carried in 3270 data. Therefore, it is possible to send graphics, IPDS™ printer data streams, and so on. The structured field consists of a structured field command and one or more

blocks following the command. However, not every TN3270 client can support all types of data. In order for clients to be able to support any of these functions, the supported range of data types needs to be determined when the Telnet connection is established. This process requires additions to TN3270. To overcome the shortcomings of traditional TN3270, TN3270 extended attributes are defined. Refer to RFC 2355 for detailed information about TN3270 enhancements (TN3270E).

In order to use the extended attributes of TN3270E, both the client and server must support TN3270E. If neither side supports TN3270E, traditional TN3270 can be used. After both sides agree to use TN3270E, they begin to negotiate the subset of TN3270E options. These options are the device-type and a set of supported 3270 functions, which are:

- Printer data stream type
- Device status information
- The passing of BIND information from server to client
- Positive/negative response exchanges

13.1.9 Device-type negotiation

Device-type names are NVT ASCII strings and all uppercase. When the TN3270E server issues the DEVICE-TYPE SEND command to the client, the server replies with a device type, a device name, or a resource name followed by the DEVICE-TYPE REQUEST command. Table 13-5 and Table 13-6 show the device-types.

Table 13-5 TN3270 device-types: Terminals

Terminal	Terminal-E	Screen size
IBM-3278-2	IBM-3278-2-E	24 row x 80 col display
IBM-3278-3	IBM-3278-3-E	32 row x 80 col display
IBM-3278-4	IBM-3278-4-E	43 row x 80 col display
IBM-3278-5	IBM-3278-5-E	27 row x 132 col display
IBM-DYNAMIC	n/a	n/a

Table 13-6 TN3270E device-type: Printer

Printer
IBM-3287-1

Archived

Because the 3278 and 3287 are commonly used devices, device-types are restricted to 3278 and 3287 terminal and printer types to simplify the negotiation. This does not mean that other types of devices cannot be used. Simply, the device-type negotiation determines the generic characteristic of the 3270 device that will be used. More advanced functions of 3270 data stream supported by the client are determined by the combination of read partition query and query reply.

The -E suffix indicates the use of extended attributes, such as partition, graphics, extended colors, and alternate character sets. If the client and the server have agreed to use extended attributes and negotiated on a device with the -E suffix, such as an IBM-DYNAMIC device or printer, both sides must be able to handle the 3270 structured field. The structured field also allows 3270 Telnet clients to issue specific 3270 data stream to host applications that the client is capable of using.

From the point of TN3270E client, it is not always possible or easy to know device names available in the network. The TN3270E server must assign the proper device to the client. This is accomplished by using a device pool that is defined on the TN3270E server. Basically, these device pools contain SNA network devices, such as terminals and printers. In other words, the TN3270E implementation maps TN3270 sessions to specific SNA logical unit (LU) names, thus effectively turning them into SNA devices. The device pool not only defines SNA network devices but also provides some other important functions for a TN3270E session. Some of these are:

- It is possible to assign one or more printers to a specific terminal device.
- It is possible to assign a group of devices to a specific organization.
- A pool can be defined that has access to only certain types of applications on the host.

The TN3270E client can issue CONNECT or ASSOCIATE commands to connect or associate the sessions to certain types of resources. However, this resource must not conflict with the definition on the server and the device-type determined during the negotiation.

13.2 Remote Execution Command protocol (REXEC and RSH)

Remote Execution Command Daemon (REXECD) is a server that allows the execution of jobs submitted from a remote host over the TCP/IP network. The client uses the REXEC or Remote Shell Protocol (RSH) command to transfer the job across to the server. Any standard output or error output is sent back to the client for display or further processing.

Principle of operation

REXECD is a server (or daemon). It handles commands issued by foreign hosts and transfers orders to subordinate virtual machines for job execution. The daemon performs automatic login and user authentication when a user ID and password are entered.

The REXEC command is used to define the user ID, password, host address, and the process to be started on the remote host. However, RSH does not require you to send a user name and password; it uses a host access file instead. Both server and client are linked over the TCP/IP network. REXEC uses TCP port 512 and RSH uses TCP port 514. See Figure 13-5 for more details.

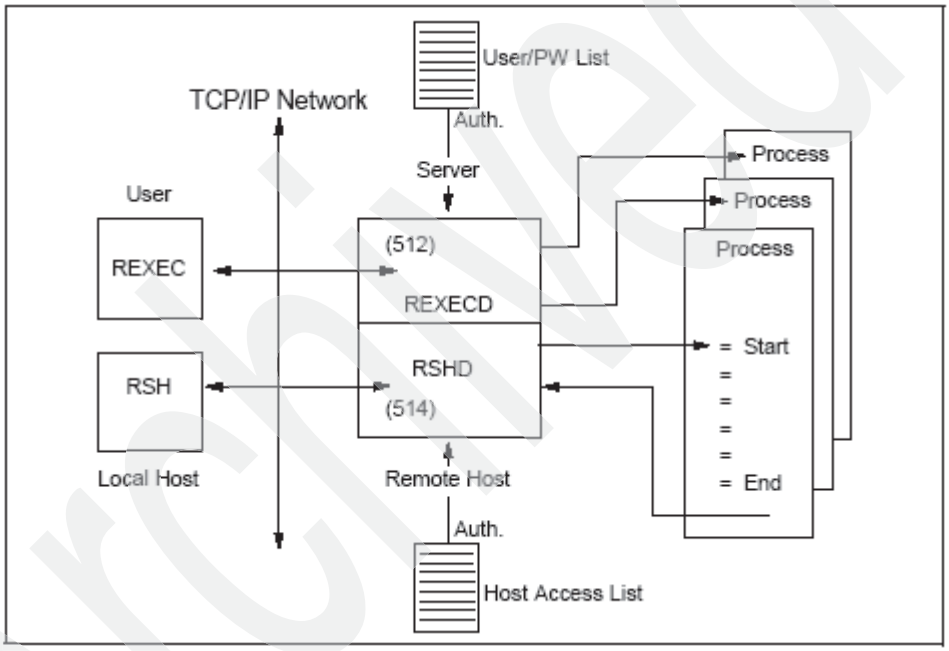


Figure 13-5 REXEC: REXECD principle

13.3 Introduction to the Distributed Computing Environment (DCE)

Distributed Computing Environment (DCE) is an architecture, a set of open standard services and associated APIs, used to support the development and administration of distributed applications in a multiplatform, multivendor environment.

DCE is the result of work from the Open Systems Foundation, or OSF (now called The Open Group), a collaboration of many hardware vendors, software vendors, clients, and consulting firms. The OSF began in 1988 with the purpose of supporting the research, development, and delivery of vendor-neutral technology and industry standards. One such standard developed was DCE. DCE Version 1.0 was released in January 1992.

As shown in Figure 13-6, DCE includes the following major services:

- Directory service
- Security service
- Distributed Time Service
- Distributed File Service
- Threads
- Remote Procedure Call

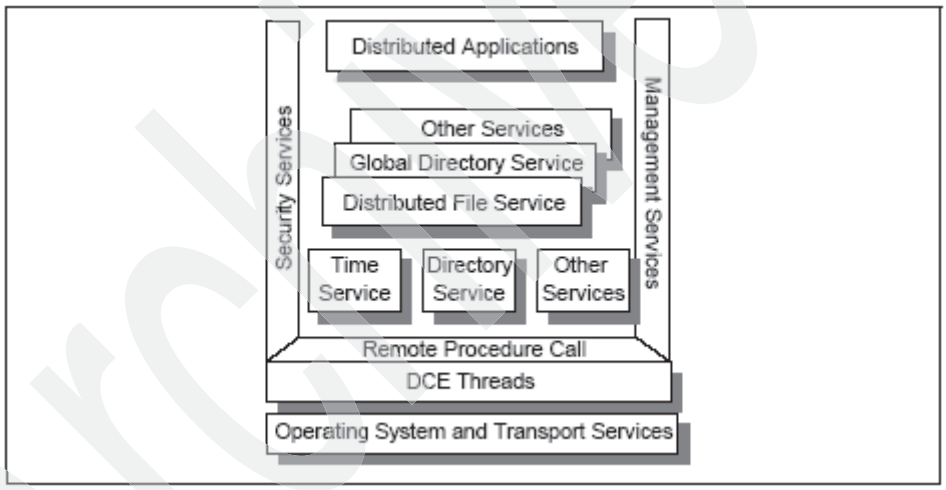


Figure 13-6 DCE architectural components

All these services have application program interfaces (APIs) that allow the programmer to use these functions. We describe these services in more detail in the following sections.

The DCE architecture does not specifically require that TCP/IP must be used for transport services, but few other protocols today meet the open and multivendor requirements of the DCE design goals. In practice, the vast majority, if not all, implementations of DCE are based on TCP/IP networks.

13.3.1 DCE directory service

When working in a large, complex network environment, it is important to keep track of the locations, names, and services (and many other details) of the participants and resources in that network. It is also important to be able to access this information easily. To enable this, information needs to be stored in a logical, central location and have standard interfaces for accessing the information. The DCE Cell Directory Service does exactly this.

The DCE directory service has the following major components:

- Cell Directory Service (CDS)
- Global Directory Service (GDS)
- Global Directory Agent (GDA)
- Application program interface (API)

Cell Directory Service

The Cell Directory Service manages a database of information about the resources in a group of closely cooperating hosts, which is called a cell. A DCE cell is very scalable and can contain many thousands of entities. Typically, even fairly large corporate companies will be organized within a single cell, which can cover several countries. The directory service database contains a hierarchical set of names, which represent a logical view of the machines, applications, users, and resources within the cell. These names are usually directory entries within a directory unit. Often, this hierarchical set of names is also called the namespace. Every cell requires at least one DCE server configured with the Cell Directory Service (a directory server).

The CDS has two very important characteristics: It can be distributed, and it can be replicated. Distributed means that the entire database does not have to reside on one physical machine in the cell. The database can logically be partitioned into multiple sections (called replicas), and each replica can reside on a separate machine. The first instance of that replica is the master replica, which has read/write access. The ability of the cell directory to be split into several master replicas allows the option of distributing the management responsibility for resources in different parts of the cell. This might be particularly important if the cell covers, say, several countries.

Each master replica can be replicated. That is, a copy of this replica can be made on a different machine (which is also a directory server). This is called a read-only replica. Read-only replicas provide both resilience and performance enhancement by allowing a host machine to perform lookups to the nearest available replica.

Replicas are stored in a clearinghouse. A clearinghouse is a collection of directory replicas at a particular server. All directory replicas must be part of a clearinghouse (although not necessarily the same one).

The Cell Directory Service makes use of the DCE security service. When the CDS initializes, it must authenticate itself to the DCE security service. This prevents a fraudulent CDS from participating in the existing cell.

Figure 13-7 shows the directory structure of the CDS namespace. As you can see, the namespace is organized in a hierarchical manner.

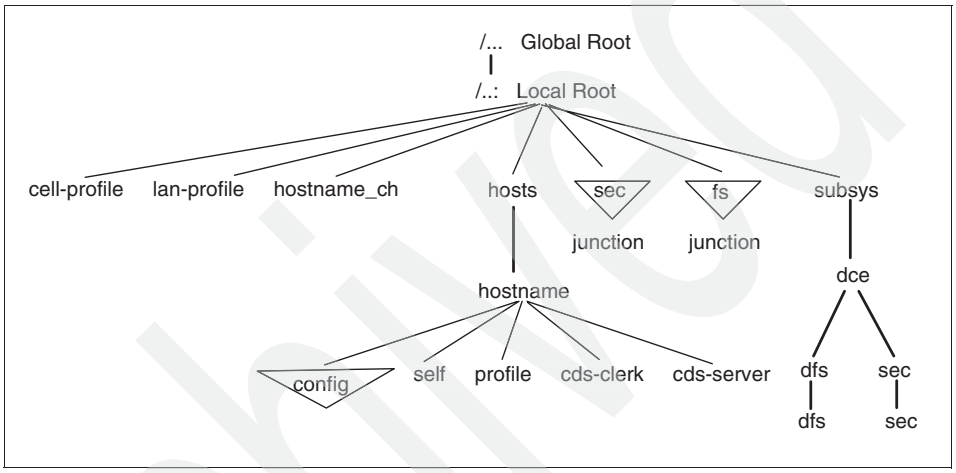


Figure 13-7 DCE: CDS namespace directory structure

Not all DCE names are stored directly in the DCE directory service. Resource entries managed by some services, such as the security service (sec) and the distributed file system (fs), connect into the namespace by means of specialized CDS entries called junctions. A junction entry contains binding information that enables a client to connect to a directory server outside of the directory service.

The security namespace is managed by the registry service of the DCE security component, and the DFS namespace is managed by the file set location database (FLDB) service of DFS.

Global Directory Service and Agent

The Cell Directory Service is responsible for knowing where resources are within the cell. However, in a multicell network, each cell is part of a larger hierarchical namespace, called the global directory namespace. The Global Directory Service (GDS) enables us to resolve the location of resources in foreign cells. This is the case when a company wants to connect their cells together or to the Internet.

In order to find a resource in another cell, a communication path needs to exist between the two cells. This communication path can currently be one of two types:

- CCITT X.500
- Internet Domain Name Services (DNS)

In order for intercell communications to be accomplished, another component, the Global Directory Agent, is required. The Global Directory Agent (GDA) is the intermediary between the local cell and the Global Directory Service. In Figure 13-8, if the CDS does not know the location of a resource, it tells the client to ask the GDA for assistance. The GDA knows to which global namespace it is connected and queries the GDS (either DNS or X.500) for the name of the foreign cell directory server with which to communicate. When in direct communication with the foreign cell directory server, the network name of the resource requested can be found. The Global Directory Agent is the component that provides communications support for either DNS or X.500 environments.

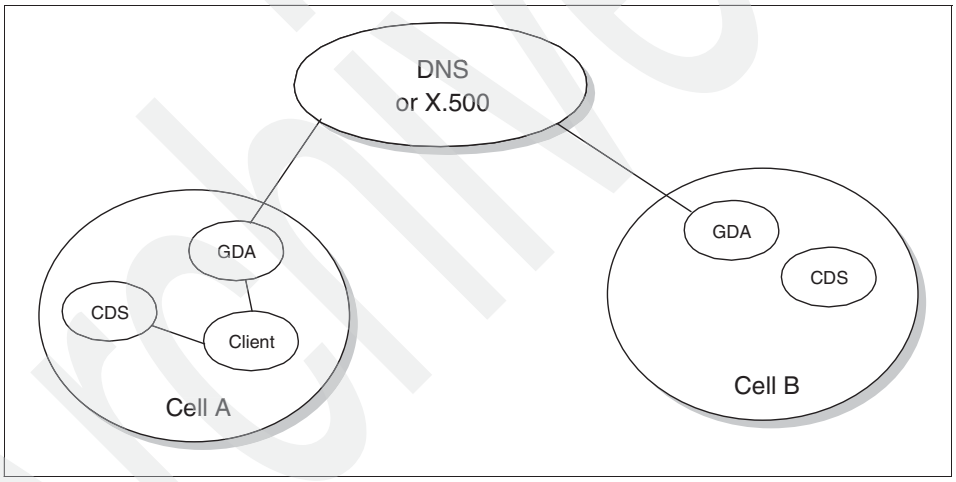


Figure 13-8 DCE: Global Directory Agent

DCE security service

Security is always a concern in a networked environment. In a large, distributed environment, it is even more crucial to ensure that all participants are valid users who access only the data with which they are permitted to work. The two primary concerns are authentication and authorization. Authentication is the process of proving or confirming the identity of a user or service. Authorization is the process of checking a user's level of authority when an access attempt is made. For example, if a user tries to make a change when read-only access has been granted, the update attempt will fail.

The DCE security service ensures secure communications and controlled access to resources in this distributed environment. It is based on the Massachusetts Institute of Technology's Project Athena, which produced Kerberos. Kerberos is an authentication service that validates a user or service. The current DCE security service (DCE 1.2.2) is based on Kerberos Version 5.

Because the DCE security service must be able to validate users and services, it must also have a database to hold this information. This is indeed the case. The DCE security service maintains a database of principals, accounts, groups, organizations, policies, properties, and attributes. This database is called the registry. Figure 13-9 shows a pictorial representation of the registry tree. The registry is actually part of the cell directory namespace, although it is stored on a separate server.

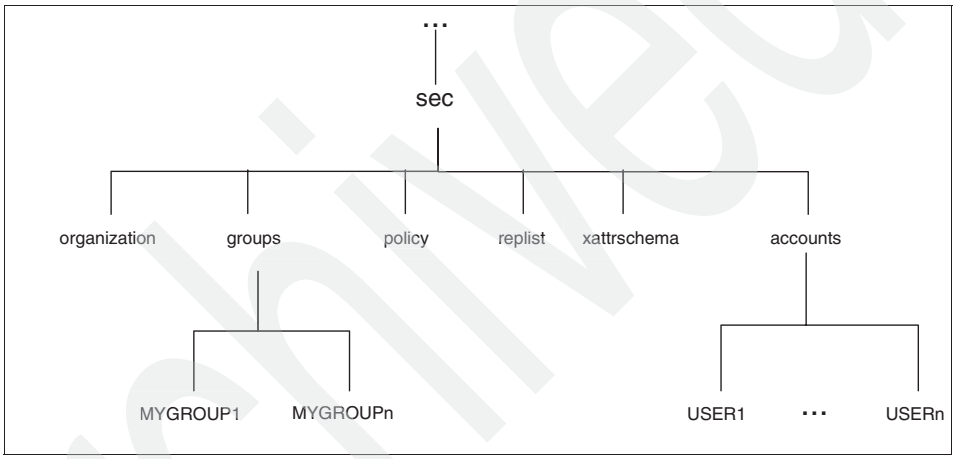


Figure 13-9 DCE: Registry directory structure

The DCE security service consists of several components:

Authentication service Handles the process of verifying that principals are correctly identified. This also contains a *ticket granting service*, which allows the engagement of secure communications.

Privilege service Supplies a user's privilege attributes to enable them to be forwarded to DCE servers.

Registry service Maintains the registry database, which contains accounts, groups, principals, organizations, and policies.

Access control list facility Provides a mechanism to match a principal's access request against the access controls for the resource.

Login facility Provides the environment for a user to log in and initialize the security environment and credentials.

These services enable user authentication, secure communication, authorized access to resources, and proper enforcement of security.

The DCE security service communicates with the Cell Directory Service to advertise its existence to the other systems that are part of the cell. The DCE security service also uses the Distributed Time Service to obtain time stamps for use in many of its processes.

13.3.2 Authentication service

The role of the authentication service is to allow principals to positively identify themselves and participate in a DCE network. Both users and servers authenticate themselves in a DCE environment, unlike security in most other client/server systems, where only users are authenticated. There are two distinct steps to authentication. At initial logon time, the Kerberos third-party protocol is used within DCE to verify the identity of a client requesting to participate in a DSS network. This process results in the client obtaining credentials, which form the basis for setting up secure sessions with DCE servers when the user tries to access resources.

In DCE Version 1.1, the idea of preauthentication was introduced, which is not present in the Kerberos authentication protocols. Preauthentication protects the security server from a rogue client trying to guess valid user IDs in order to hack into the system. In DCE 1.1, there are three protocols for preauthentication:

No preauthentication This is provided to support DCE clients earlier than Version 1.1.

Timestamps This is used by DCE Version 1.1 clients that are unable to use the third-party protocol. An encrypted time stamp is sent to the security server. The time stamp is decrypted, and if the time is within five minutes, the user is considered preauthenticated. This option needs to be specified for cell administrators and non-interactive principals.

Third-party This is the default used by DCE Version 1.1 (and later) clients. It is similar to the time stamps protocol, but additional information about the client is also encrypted in various keys.

The login and authentication process using the third-party preauthentication protocol is shown in Figure 13-10.

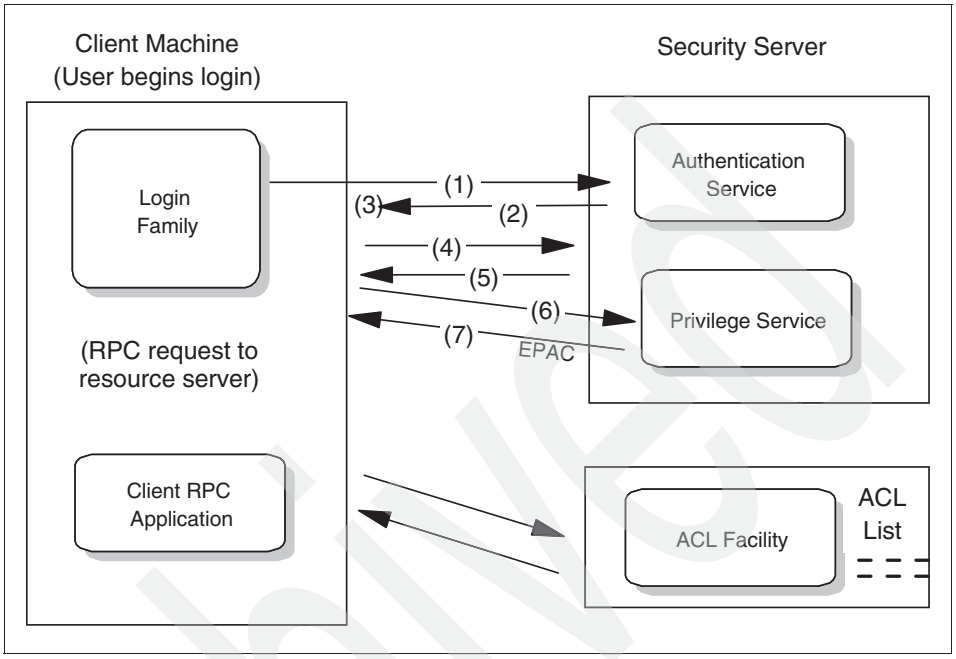


Figure 13-10 DCE: Authentication and login process using third-party protocol

This detailed process is as follows:

1. The user issues a request to log in to the cell. However, the user must first be authenticated. The client creates two random conversation keys, one of them based on the machine session key. The login facility on the client then uses these keys and the supplied password to encrypt a request for an authentication ticket (ticket granting ticket, or TGT) from the security server.
2. The authentication service (AS) on a security server receives the request. The AS looks up the machine session key from the registry to decrypt the request. (Note that by knowing the machine session key, the security server proves to be valid for the cell. A false server would not know the machine session key.) If the decryption is successful and the time stamp is within five minutes, the AS encrypts a TGT using one of the client conversation keys provided. When successful, this encrypted TGT is returned to the client.
3. The client receives the TGT envelope and decrypts it using one of the client conversation keys it provided. Also included is a conversation key for the AS. Note that the TGT itself is encrypted with a conversation key of the AS. This valid TGT is proof that the user is now authenticated.

4. Now the user needs the authorization credentials, known as extended privilege attribute certificate (EPAC), from the privilege service (PS). Therefore, it must construct a privilege ticket granting ticket (PTGT) request to retrieve this from the PS. To communicate with the PS, the client sends a request to the AS to contact the PS. This request is encrypted with the conversation key of the AS.
5. The AS receives this request. Using the secret key of the PS, the AS generates a conversation key for the client to use when contacting the PS. This is returned to the client and encrypted again with the AS conversation key. The client receives the envelope and decrypts it (using the conversation key) and discovers the conversation key for the PS. The client can now send a privilege service ticket to the PS.
6. The PS receives the request and decrypts it with its secret key successfully. This proves that the service ticket is legitimate, which also implies that the AS involved is also legitimate. From this, the PS knows that the client and the AS are valid. The PS constructs the EPAC, which lists the user's standard and extended registry attributes, including group membership. The PS creates more conversation keys and sends the EPAC and other information in an encrypted PTGT envelope to the client.
7. The client decrypts the PTGT envelope using the PS conversation key. Also, the client has the conversation key information and an encrypted PTGT (which the client cannot decrypt, because it is encrypted using the AS secret key).
8. Now, the client wants to contact an application server. To do so, it sends the PTGT to the AS and requests a service ticket for the application server. The AS receives the PTGT and decrypts it to obtain the EPAC information. It encrypts the EPAC information with the secret key of the application server and also provides a conversation key for the application server. This information is encrypted with the conversation key of the AS (which the client knows) and is returned to the client.
9. The client decrypts the envelope and discovers the application server's secret conversation key. Using this key, it can now contact the application server. By correctly decrypting the request from the client, the application server is able to determine that the client has been authenticated, and by responding to the client, the client knows that it was, indeed, the real application server that it has contacted. The two will then establish a *mutually authenticated* session.

In addition to the extensive use of secret keys during the logon process, third-party authentication makes use of time stamps to ensure that the conversation is protected against intruders and eavesdropping. Time stamps make impersonation techniques, such as record and playback, ineffective. Also, the actual user password entered at logon time does not flow to the server as such. Instead, it is used as an encryption key for the initial logon messages that

are then decrypted by the security server using its own copy of the password stored in the registry database.

If the security server is not able to authenticate the client for some reason, such as the entering an invalid password, an error is returned and the logon is terminated. However, if the exchange completes with the client being successfully authenticated, the security server returns credentials that are then used by the client to establish sessions with other DCE servers, such as resource and directory servers. These credentials contain information in the form of a privilege ticket granting ticket (PTGT) and extended privilege attribute certificate (EPAC):

EPAC This is a validated list supplied by the security server containing the client's name, groups the client belongs to, and the extended registry attributes for the authenticated client (if any were defined and associated with their account). A client must present its EPAC (acquired during third-party authentication) to any server the client wants to connect to in order to access its resources.

PTGTA PTGT is a privilege ticket granting ticket. It contains the EPAC, which has all the relevant information about a user (UUID, group membership, ERAs, and so on). The PTGT is what is actually passed from a DCE client to a DCE server when it needs to access resources.

Public key support

The latest version of DCE (DCE Version 1.2.2) introduces the option of using public key technology (such as that from RSA or smart cards) to support the login process. Using this technology, the long-term key (or password) for a user (or other DCE object) does not need to be stored at the security server, providing enhanced security in the event of a compromise of the security server. Administrators can specify that some principals can use the pre-DCE 1.2 mechanisms while others have access to the public key mechanism. DCE_1.2.2 retains full interoperability with previous DCE releases. During the login process, public key users receive credentials that allow them to use the current (Kerberos-based) DCE authentication mechanism. A new pre-authentication protocol is used. The login client does not have to determine whether a given user is public key-capable prior to requesting credentials.

13.3.3 DCE threads

Traditional applications (written in languages such as C, COBOL, and so on) have many lines of programming code that usually execute in a sequential manner. At any time, there is one point in the program that is executing. This can

be defined as single threading. A thread is a single unit of execution flow within a process. Better application performance can often be obtained when a program is structured so that several areas can be executed concurrently. This is called multithreading. The capability of executing multiple threads also depends on the operating system.

In a distributed computing environment based on the client/server model, threads provide the ability to perform many procedures at the same time. Work can continue in another thread while the thread waiting for a specific response is blocked (for example, waiting for response from the network). A server can issue concurrent procedure call processing. While one server thread is waiting for an I/O operation to finish, another server thread can continue working on a different request.

To function well, thread support needs to be integrated into the operating system. If threads are implemented at the application software level instead of within the operating system, performance of multithreaded applications might seem slow.

The DCE thread APIs are either user-level (in operating systems that do not support threads, such as Microsoft Windows® 3.x) or kernel threads (such as IBM AIX® 5L™ and OS/2®). They are based on the POSIX 1003.4a Draft 4 standard. Because OS/2 also has threads, the programmer can use DCE threads or OS/2 threads.

DCE threads can be *mapped* onto OS/2 threads through special programming constructs. However, in order to write portable applications that can run on different platforms, only DCE threads must be used. In many cases, there is little performance difference resulting from this mapping.

DCE Remote Procedure Call

The DCE Remote Procedure Call (RPC) architecture is the foundation of communication between the client and server in the DCE environment.

Note: The DCE RPC is conceptually similar to the ONC RPC (see 11.2.2, “Remote Procedure Call (RPC)” on page 415), but the protocol used on the wire is not compatible.

RPCs provide the ability for an application program's code to be distributed across multiple systems, which can be anywhere in the network.

An application written using DCE RPCs has a client portion, which usually issues RPC requests, and a server portion, which receives RPC requests, processes them, and returns the results to the client. RPCs have three main components:

- The Interface Definition Language (IDL) and its associated compiler. From the specification file, it generates the header file, the client stub, and the server stub. This allows an application to issue a Remote Procedure Call in the same manner as it issues a local procedure call.
- The network data representation, which defines the format for passing data, such as input and output parameters. This ensures that the bit-ordering and platform-specific data representation can be converted properly after it arrives at the target system. This process of preparing data for an RPC call is called marshalling.
- The runtime library, which shields the application from the details of network communications between client and server nodes.

The application programmer can choose to use multiple threads when making RPC calls. This is because an RPC is synchronous; that is, when an RPC call is made, the thread that issued the call is blocked from further processing until a response is received.

Remote Procedure Calls can be used to build applications that make use of other DCE facilities, such as the Cell Directory Service (CDS) and the security service. The CDS can be used to find servers or to advertise a server's address for client access. The security service might be used to make authenticated RPCs that enable various levels of data integrity and encryption using the commercial data masking facility (CDMF), data encryption standard (DES), and other functions such as authorization.

13.3.4 Distributed Time Service

Keeping the clocks on different hosts synchronized is a difficult task because the hardware clocks do not typically run at the same rates. This presents problems for distributed applications that depend on the ordering of events that happen during their execution. For example, let us say that a programmer is compiling some code on a workstation and some files are also located on a server. If the workstation and the server do not have their time synchronized, it is possible that the compiler might not process a file, because the date is older than an existing one on the server. In reality, the file is newer, but the clock on the workstation is slow. As a result, the compiled code will not reflect the latest source code changes. This problem becomes more acute in a large cell where servers are distributed across multiple time zones.

The DCE Distributed Time Service (DTS) provides standard software mechanisms to synchronize clocks on the different hosts in a distributed

environment. It also provides a way of keeping a host's time close to the absolute time. DTS is optional. It is not a required core service for the DCE cell. However, if DTS is not implemented, the administrator must use some other means of keeping clocks synchronized for all the systems in the cell.

The Distributed Time Service has several components. They are:

- Local time server
- Global time server
- Courier and backup courier time server

Local time server

The local time server is responsible for answering time queries from time clerks on the LAN. Local time servers also query each other to maintain synchronization on the LAN. If a time clerk cannot contact the required number of local time servers (as specified by the minservers attribute), it must contact global time servers through a CDS lookup.

We recommend that there are at least three local time servers per LAN. This ensures that the time on the LAN is synchronized. The task of synchronization across multiple LANs in the cell is performed by global and courier time servers.

Global time server

A global time server (GTS) advertises itself in the Cell Directory Service namespace so that all systems can find it easily. A GTS participates in the local LAN in the same way that local time servers do, but it has an additional responsibility. It also gives its time to a courier time server, which is located in a different LAN.

Courier roles

Local and global time servers can also have a courier role. They can be couriers, backup couriers, or non-couriers. The courier behaves similarly to other local time servers, participating in the time synchronization process. However, the courier does not look at its own clock. It requests the time from a global time server located in another LAN or in another part of the cell. Because the time is imported from another part of the network, this enables many remote LAN segments in all parts of the cell to have a very closely synchronized time value.

The backup courier role provides support in the event that the primary courier for that LAN is not available. The backup couriers will negotiate to elect a new courier and thus maintain the proper time synchronization with the global time servers. Note that even if a courier time server is not defined, local time servers and clerks will try to contact a global time server if they cannot contact the minimum number of servers from the local segment.

The default for time servers is the non-courier role. As long as enough local time servers can be contacted, they will not contact a global time server.

In a large or distributed network, local time servers, global time servers, and courier time servers automatically and accurately make the process of time synchronization function.

13.3.5 Additional information

For additional information about DCE, refer to the IBM Redbook *DCE Cell Design Considerations*, SG24-4746.

For information about the most current release of DCE (Version 1.2.2), view the Open Group Web site at:

`http://www.opengroup.org/dce`

13.4 Distributed File Service (DFS)

The Distributed File Service is not really a core component of DCE, but it is an application that is integrated with, and uses, the other DCE services. DFS provides global file sharing. Access to files located anywhere in interconnected DCE cells is transparent to the user. To the user, it appears as though the files were located on a local drive. DFS servers and clients can be heterogeneous computers running different operating systems.

The origin of DFS is Transarc Corporation's implementation of the Andrew File System (AFS) from Carnegie-Mellon University. DFS conforms to POSIX 1003.1 for file system semantics and POSIX 1003.6 for access control security. DFS is built onto, and integrated with, all of the other DCE services, and was developed to address identified distributed file system needs, such as:

- Location transparency
- Uniform naming
- Good performance
- Security
- High availability
- File consistency control
- NFS interoperability

Archived

DFS follows the client/server model, and it extends the concept of DCE cells by providing DFS administrative domains, which are an administratively independent collection of DFS server and client systems within a DCE cell.

There can be many DFS file servers in a cell. Each DFS file server runs the file exporter service that makes files available to DFS clients. The file exporter is also known as the protocol exporter. DFS clients run the cache manager, which is an intermediary between applications that request files from DFS servers. The cache manager translates file requests into RPCs to the file exporter on the file server system and stores (caches) file data on disk or in memory to minimize server accesses. It also ensures that the client always has an up-to-date copy of a file.

The DFS file server can serve two different types of file systems:

- Local file system (LFS), also known as the episode file system
- Some other file system, such as the UNIX File System (UFS)

Full DFS functionality is only available with LFS and includes:

- High performance
- Log-based, fast restarting file sets for quick recovery from failure
- High availability with replication, automatic updates, and automatic bypassing of failed file server
- Strong security with integration to the DCE security service providing ACL authorization control

13.4.1 File naming

DFS uses the Cell Directory Service (CDS) name /.:/fs as a junction to its self-administered namespace. DFS objects of a cell (files and directories) build a file system tree rooted in /.:/fs of every cell. Directories and files can be accessed by users anywhere in the network using the same file or directory names, no matter where they are physically located, because all DCE resources are part of a global namespace.

As an example of DFS file naming, to access a particular file from within a cell, a user might use the following name:

```
/.:/fs/usr/woodd/games/tictactoe.exe
```

From outside the cell, using GDS (X.500) format, the following name is used:

```
/.../C=US/O=IBM/OU=ITSC/fs/usr/woodd/games/tictactoe.exe
```

Or, in DNS format:

```
/.../itsc.ibm.com/usr/woodd/games/tictactoe.exe
```

13.4.2 DFS performance

Performance is one of the main goals of DFS, and it achieves it by including features such as:

Cache manager Files requested from the server are stored in cache at the client so that the client does not need to send requests for data across the network every time the user needs a file. This reduces load on the server file systems and minimizes network traffic, thereby improving performance.

Multithreaded servers DFS servers make use of DCE threads support to efficiently handle multiple file requests from clients.

RPC pipes The RPC pipe facility is extensively used to transport large amounts of data efficiently.

Replication Replication support allows efficient load-balancing by spreading out the requests for files across multiple servers.

File consistency

Using copies of files cached in memory at the client side can potentially cause problems when the file is being used by multiple clients in different locations. DFS uses a token mechanism to synchronize concurrent file accesses by multiple users and ensure that each user is always working with the latest version of a file. The whole process is transparent to the user.

Availability

LFS file sets can be replicated on multiple servers for better availability. Every file set has a single read/write version and multiple read-only replicas. The read/write version is the only one that can be modified. Every change in the read/write file set is reflected in the replicated file sets. If there is a crash of a server system housing a replicated file set, the work is not interrupted, and the client is automatically switched to another replica.

DFS security

DCE security provides DFS with authentication of user identities, verification of user privileges, and authorization control. Using the DCE security's ACL mechanism, DFS provides more flexible and powerful access control than that

typically provided by an operating system (for example, UNIX read, write, and execute permissions).

DFS/NFS interoperability

DFS files can be exported to NFS so that NFS clients can access them as unauthenticated users. This requires an NFS/DFS authenticating gateway facility, which might not be available in every implementation.

13.5 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 854 – TELNET Protocol Specifications (May 1983)
- RFC 855 – TELNET Option Specifications. (May 1983)
- RFC 2355 – TN3270 Enhancements (June 1998)

Archived

14

File-related protocols

The TCP/IP protocol suite provides a number of protocols for the manipulation of files. In general, there are two different mechanisms for accessing remote files. The most simple mechanism is by transferring the particular file to the local machine. In this case, multiple copies of the same file are likely to exist. File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Secure Copy (SCP), and SSH FTP (SFTP) employ this mechanism of file sharing.

An alternate approach to accessing files is through the use of a file system. In this case, the operating machine on the local host provides the necessary functionality to access the file on the remote machine. The user and application on the local machine are not aware that the file actually resides on the remote machine; they just read and write the file through the file system as though it were on the local machine. In this case, only one copy of the file exists, and the file system is responsible for coordinating updates. The Network File System (NFS), Andrew File System (AFS), and the Common Internet File System (CIFS, previously called Server Message Block, or SMB) provide this type of functionality.

© Copyright IBM Corp. 1989-2006. All rights reserved.

14.1 File Transfer Protocol (FTP)

FTP is a standard protocol with STD Number 9. Its status is recommended. It is described in RFC 959 – File Transfer Protocol (FTP) and updated by RFCs 2228 (FTP Security Extensions), 2428 (FTP Extensions for IPv6 and NATs), and 4217 (Securing FTP with TLS). Additional information is in RFC 2577 (FTP Security Considerations).

Transferring data from one host to another is one of the most frequently used operations. Both the need to upload data (transfer data from a client to a server) and download data (retrieve data from a server to a client) are addressed by FTP. Additionally, FTP provides security and authentication measures to prevent unauthorized access to data.

14.1.1 An overview of FTP

FTP uses TCP as a transport protocol to provide reliable end-to-end connections and implements two types of connections in managing data transfers. The FTP client initiates the first connection, referred to as the control connection, to well-known port 21 (the client's port is typically ephemeral). It is on this port that an FTP server listens for and accepts new connections. The control connection is used for all of the control commands a client user uses to log on to the server, manipulate files, and terminate a session. This is also the connection across which the FTP server will send messages to the client in response to these control commands.

Note: Communication between the client and the server follow the Telnet protocol defined in RFC 854.

The second connection used by FTP is referred to as the data connection. Typically, the data connection is established on server port 20. However, depending on how the data connection is established, both the client and server might use ephemeral ports. It is across this connection that FTP transfers the data. FTP only opens a data connection when a client issues a command requiring a data transfer, such as a request to retrieve a file, or to view a list of the files available. Therefore, it is possible for an entire FTP session to open and close without a data connection ever having been opened. Unlike the control connection, in which commands and replies can flow both from the client to the server and from the server to the client, the data connection is unidirectional. FTP can transfer data only from the client to the server, or from the server to the client, but not both. Also, unlike the control connection, the data connection can be initiated from either the client or the server. Data connections initiated by the server are active, while those initiated by the client are passive.

The client FTP application is built with a protocol interpreter (PI), a data transfer process (DTP), and a user interface. The server FTP application typically only consists of a PI and DTP (see Figure 14-1).

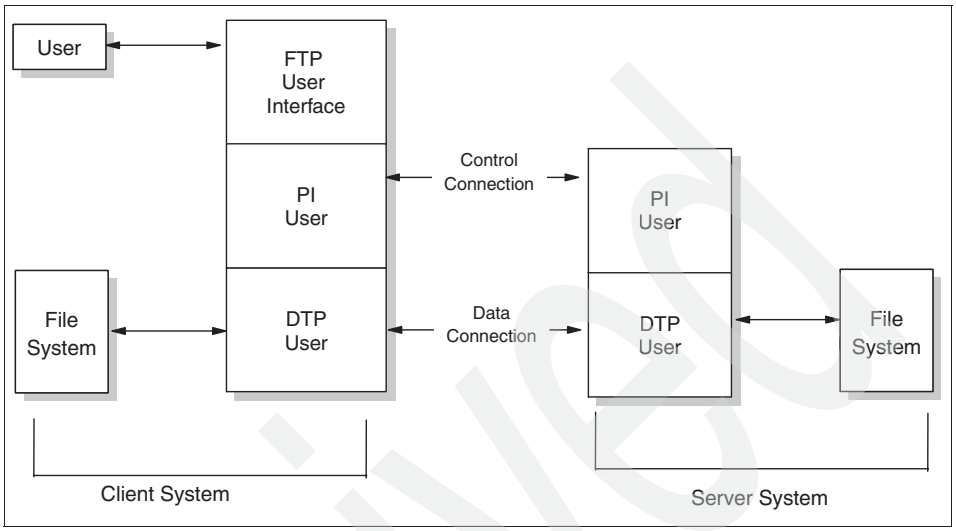


Figure 14-1 The FTP model

The FTP client's user interface communicates with the protocol interpreter, which manages the control connection. This protocol interpreter translates any application-specific commands to the RFC architected FTP commands, and then communicates these control commands to the FTP server.

The FTP server's PI receives these commands, and then initiates the appropriate processes to service the client's requests. If the requests require the transfer of data, data management is performed by the DTPS on both the client and server applications. Following the completion of the data transfer, the data connection is closed, and control is returned to the PIs of the client and server applications.

Note: Only one data transfer can occur for each data connection. If multiple data transfers are required for a single FTP session, one distinct control connection will be opened for each transfer.

14.1.2 FTP operations

When using FTP, the user performs some or all of the following operations:

- Connect to a remote host.

- Navigate and manipulate the directory structure.
- List files available for transfer.
- Define the transfer mode, transfer type, and data structure.
- Transfer data to or from the remote host.
- Disconnect from the remote host.

Connecting to a remote host

To execute a file transfer, the user begins by logging in to the remote host. This is only the primary method of implementing security within the FTP model. Additional security can be provided using SSL and TLS, which we discuss later in this chapter (see 14.1.9, “Securing FTP sessions” on page 527). Conversely, this authentication can be bypassed using anonymous FTP (see 14.1.7, “Anonymous FTP” on page 525).

There are four commands that are used:

`open`	Selects the remote host and initiates the login session.
`user`	Identifies the remote user ID.
`pass`	Authenticates the user.
`site`	Sends information to the foreign host that is used to provide services specific to that host.

Navigating the directory structure

After a user has been authenticated and logged on to the server, that user can navigate through the directory structure of the remote host in order to locate the file desired for retrieval, or locate the directory into which a local file will be transferred. The user can also navigate the directory structure of the client’s host. After the correct local and remote directories have been accessed, users can display the contents of the remote directory. The subcommands that perform these functions are as follows:

`cd`	Changes the directory on the remote host: A path name can be specified, but must conform to the directory structure of the remote host. In most implementations, **`cd ..`** will move one directory up within the directory structure.
`lcd`	Changes the directory on the local host. Similar to the **`cd`** command, a path name can be specified but must conform to the directory structure of the local host.

ls Lists the contents of the remote directory. The list generated by this command is treated as data, and therefore, this command requires the use of a data connection. This command is intended to create output readable by human users.

dir Lists the contents of the remote directory. Similar to the ls command, the list generated by dir is treated as data and requires the use of a data connection. This command is intended to create output readable by programs.

Controlling how the data is transferred

Transferring data between dissimilar systems often requires transformations of the data as part of the transfer process. The user has to decide on three aspects of the data handling:

- The way the bits will be moved from one place to another
- The different representations of data on the system's architecture
- The file structure in which the data is to be stored

Each of these is controlled by a subcommand:

mode Specifies whether the file is treated as having a record structure in a byte stream format:

- **B** This specifies block mode is to be used. This indicates that the logical record boundaries of the file are preserved.
- **S** This specifies that stream mode is to be used, meaning that the file is treated as a byte stream. This is the default and provides more efficient transfer but might not produce the desired results when working with a record-based file system.

type Specifies the character sets used in translating and representing the data:

- **A** Indicates that both hosts are ASCII-based, or that if one is ASCII-based and the other is EBCDIC-based, that ASCII-EBCDIC translation must be performed. On many implementations, this can be invoked by issuing the ASCII command, which the PI translates into the **type** a command.
- **E** Indicates that both hosts use an EBCDIC data representation. On many implementations, this can be invoked by issuing the EBCDIC command, which the PI translates into the **type E** command.

Archived

I Indicates that no translation is to be done on the data. On many implementations, this can be invoked by using the BINARY command, which the PI translates into the **type I** command.

structure Specifies the structure of the file to be transferred:

file Indicates that the file has no internal structure, and is considered to be a continuous sequence of data bytes.

record Indicates that the file is made up of sequential records.

page Indicates that the file is made up of independent indexed pages.

Note: RFC 959 minimum FTP implementation includes only the **file** and **record** options for the **structure** command. Therefore, not all implementations will include the **page** option.

Because these subcommands do not cover all possible differences between systems, the **site** subcommand is available to issue implementation-dependent and platform-specific commands. The syntax of this command will vary by implementation.

Transferring files

The following commands can be used to copy files between FTP clients and servers:

get Copies a file from the remote host to the local host. The PI translates **get** into a RETR command.

mget Copies multiple files from the remote to the local host. The PI translates **mget** into a series of RETR commands.

put Copies a file from the local host to the remote host. The PI translates **put** into a STOR command.

mput Copies multiple files from the local host to the remote host. The PI translates **mput** into a series of STOR commands.

Terminating the FTP session

The following commands are used to end an FTP session:

quit Disconnects from the remote host and terminates FTP. Some implementations use the BYE subcommand.

close Disconnects from the remote host but leaves the FTP client running. An **open** command can be issued to establish a new control connection.

An example of an FTP scenario

A LAN user has to transfer data from a workstation to the remote host remote.host. The data is binary, and exists in a file structure named mydata in the workstation's /localfolder directory. The user wants to transfer the data, using stream mode, to the remote host's /tmp directory, and would like the new file to be named yourdata. The process for doing this is illustrated in Figure 14-2.

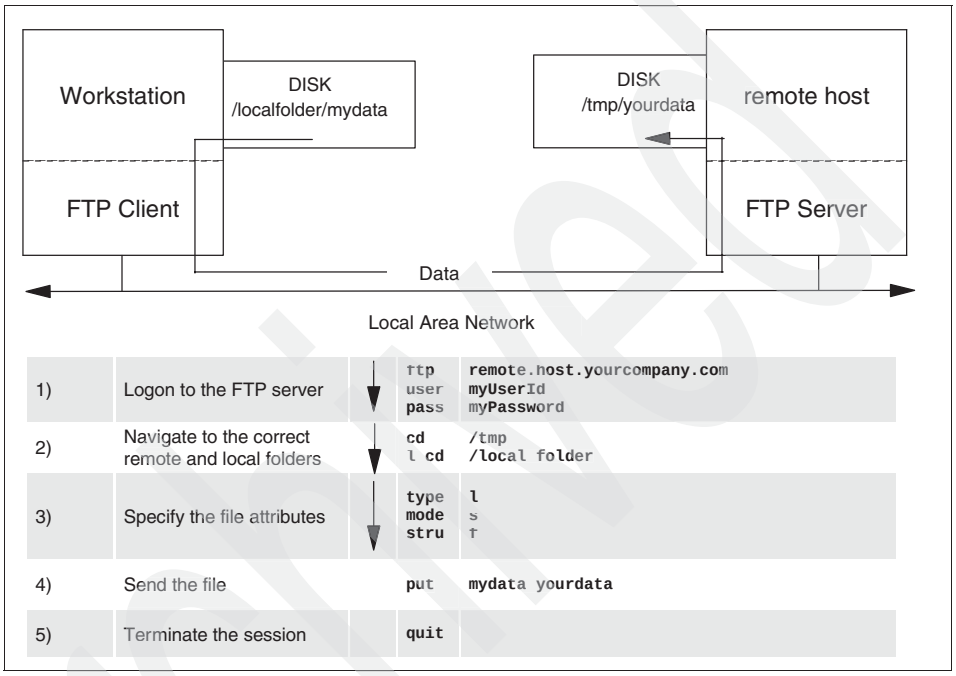


Figure 14-2 An example of an FTP transfer

Archived

14.1.3 The active data transfer

When issuing the subcommands **get**, **put**, **mget**, **mput**, **ls**, or **dir**, FTP opens a data connection across which the data will be transferred. Most FTP implementations default to an active data transfer unless a passive transfer has been specifically requested. In an active transfer, the FTP client sends a PORT command to the FTP server, indicating the IP address and port number on which the client will listen for a connection. Upon accepting the PORT command, the FTP server initiates a connection back to the client on the indicated IP address and port. After this connection has been established, data will begin flowing, either from the client to the server (for **put** and **mput** commands) or from the server to the client (for **get**, **mget**, **ls**, and **dir** commands). An example of this sequence is illustrated in Figure 14-3.

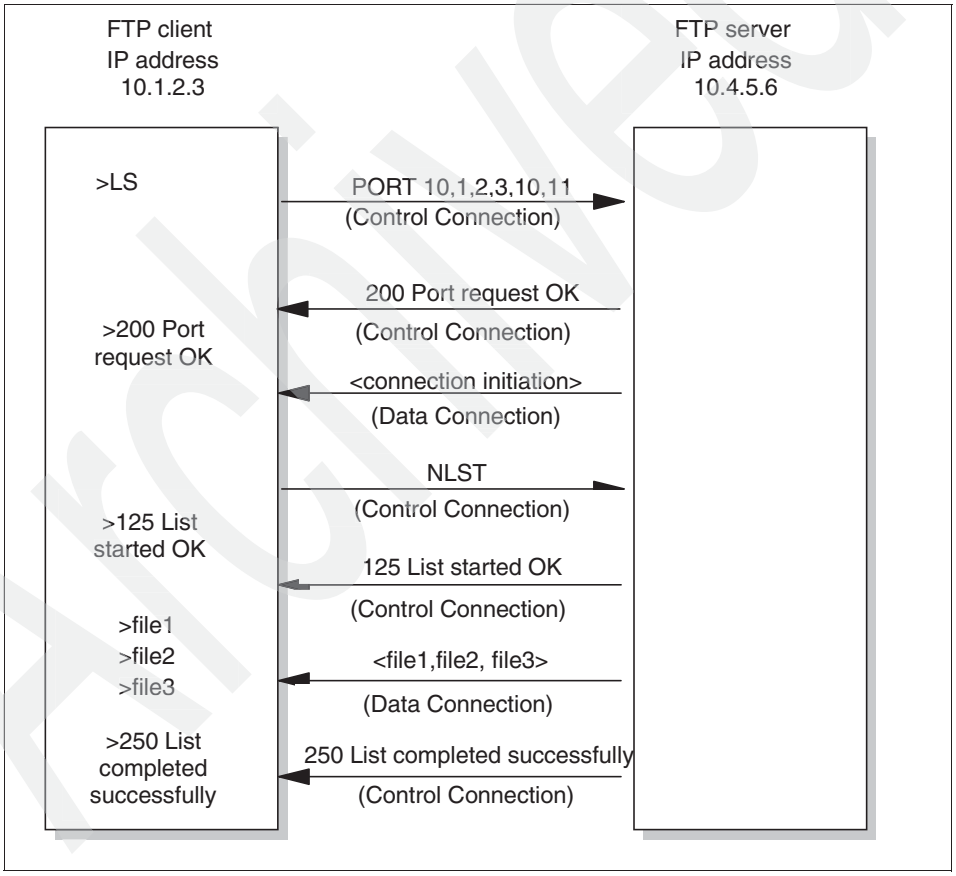


Figure 14-3 The active data connection

Archive

14.1.4 The passive data transfer

Contrary to the use of an active data connection, the passive data transfer reverses the direction of establishment of the data connection. Instead of issuing a PORT command, the client issues a PASV command, which uses no parameters. Upon accepting this command, the FTP server sends back a reply containing an IP address and port number. The client initiates a connection back to the server on the indicated IP address and port. An example of this sequence is illustrated in Figure 14-4.

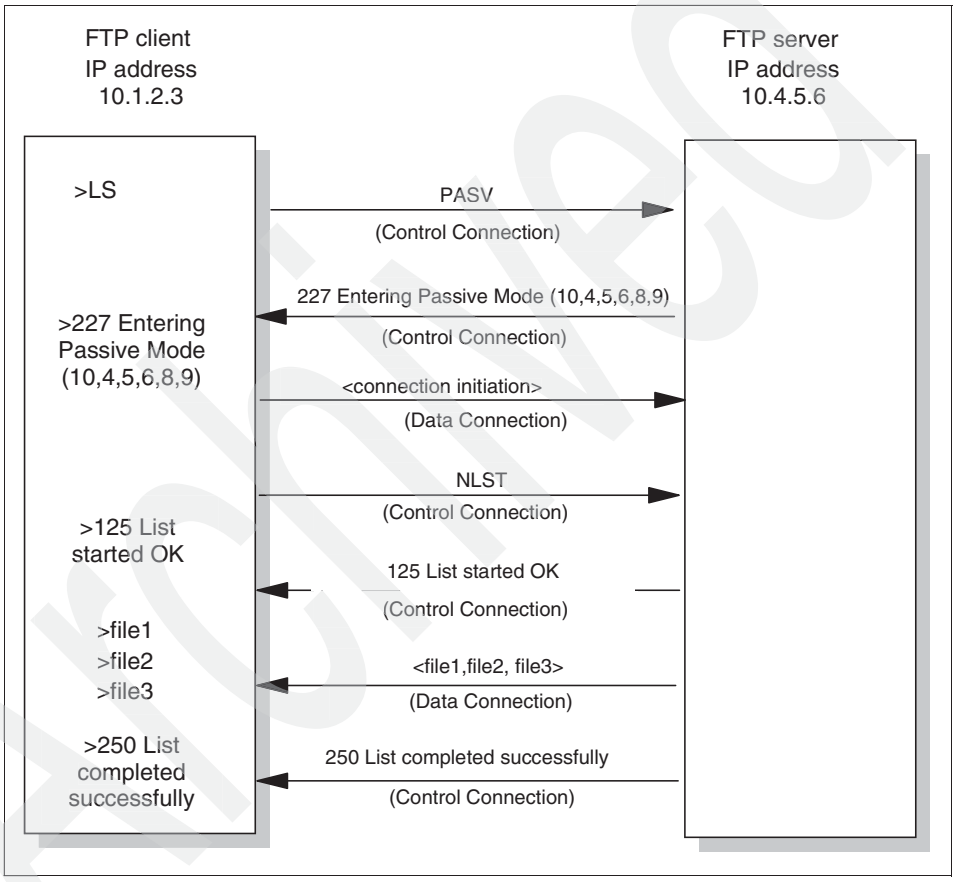


Figure 14-4 The passive data connection

One of the reasons to use a passive data transfer is to bypass firewall configurations that block active data connections. For this reason, passive mode is often referred to as “firewall friendly mode.” An example of such a scenario is a firewall that has been configured to block any inbound attempts to open a connection. In this example, an FTP server responding to a client’s PORT

command would receive an error when trying to open a connection to the indicated IP address and port. However, by using passive mode, the client initiates the connection from within the network, and the firewall allows the data transfer to proceed. Other methods exist to resolve problems when FTPing through a firewall, including proxy transfers (see 14.1.5, “Using proxy transfer” on page 522) and the use of EPSV (see 14.1.8, “Using FTP with IPv6” on page 525).

14.1.5 Using proxy transfer

FTP provides the ability for a client to have data transferred from one FTP server to another FTP server. Several justifications for such a transfer exist, including:

- To transfer data from one host to another when direct access to the two hosts are not possible
- To bypass a slow client connection
- To bypass a firewall restriction
- To reduce the amount of traffic within the client’s network

The process of setting up a proxy transfer begins with the use of a **proxy open** command. Any FTP command can then be sent to the proxy server by preceding the command with **proxy**. For example, executing the **dir** command lists the files on the primary FTP server. Executing the **proxy dir** command lists the files on the proxy server. The **proxy get** and **proxy put** commands can then be used to transfer data between the two hosts. This process is illustrated in Figure 14-5 on page 523.

Archived

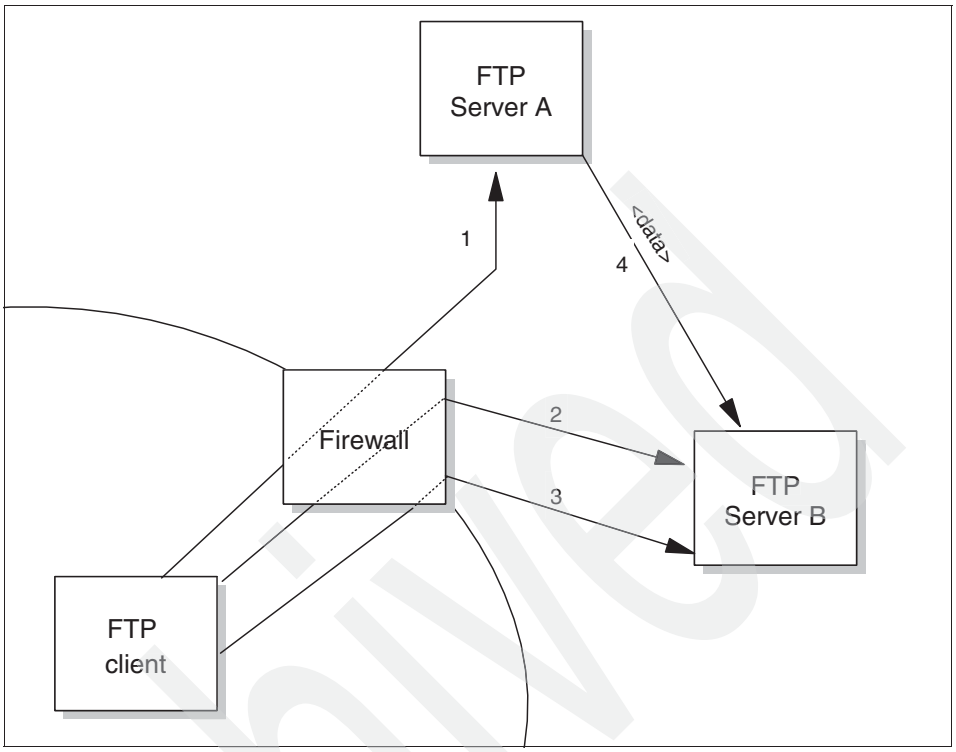


Figure 14-5 An FTP proxy transfer through a firewall

In Figure 14-5:

1. The FTP client opens a connection and logs on to the FTP server A.
2. The FTP client issues a **proxy open** command, and a new control connection is established with FTP server B.
3. The FTP client then issues a **proxy get** command (though this can also be a **proxy put**).
4. A data connection is established between server A and server B. Following data connection establishment, the data flows from server B to server A.

14.1.6 Reply codes

In order to manage these operations, the client and server conduct a dialog using the Telnet convention. The client issues commands, and the server responds with reply codes. The responses also include comments for the benefit of the user, but the client application uses only the codes. Reply codes are three digits

long, with the first two digits having specific meanings. See Table 14-1 and Table 14-2 for descriptions of these codes.

Table 14-1 First digit reply codes and descriptions

First digit reply code	Description
1yz	Positive preliminary reply. These messages are usually informational, and typically are followed by a message starting with a different reply code.
2yz	Positive completion reply. This indicates that the last client command has completed successfully.
3yz	Positive intermediate reply. This indicates that the client's command has been accepted, but that additional information from the client is needed before the command can complete.
4yz	Transient negative completion reply. This indicates that the client's command has failed, but that retrying the command might yield a success.
5yz	Permanent negative completion reply. This indicates that the command has failed, and that retrying the command will not yield a success.
6yz	Protected reply. This indicates that the reply contains security-related information.

Table 14-2 Second digit reply codes and descriptions

Second digit reply code	Description
x0z	Syntax error. This indicates that a syntax error was encountered when processing the client's command.
x1z	Requested information. This indicates that the message contains information requested by the client.
x2z	Connection information. This indicates that the reply contains information about the control or data connection.
x3z	Authentication. This indicates that the reply is part of the logon sequence, or some other authentication sequence (such as TLS authentication).
x4z	Not in use.
x5z	File system information. This indicates that the reply contains file system information relevant to the last client command received.

For each user command, shown as **this**, the FTP server responds with a 3-digit reply and message, shown as *this*:

FTP foreignhost
220 service ready
USERNAME cms01
331 user name okay
PASSWORD xyxyx
230 user logged in
TYPE Image
200 command okay

14.1.7 Anonymous FTP

Many TCP/IP sites implement what is known as anonymous FTP, which means that these sites allow public access to some file directories. The remote user only needs to use the login name *anonymous* and password *guest* or some other common password conventions (for example, the user's Internet e-mail ID). The password convention used on a system is explained to the user during the login process. Because this method of logon is available to anyone with Internet access to the system, most hosts restrict the folders accessible to anonymous users.

14.1.8 Using FTP with IPv6

The increasing use of IP version 6 (IPv6) in networks has little effect on FTP implementations, because IPv6 addressing is managed by the IP layer (see Chapter 9, “IP version 6” on page 327). However, a problem does arise with the previously architected PORT command. To understand this, we must discuss the PORT command in greater detail.

As discussed previously (see 14.1.3, “The active data transfer” on page 520), the FTP client uses the PORT command to inform the server of what port and IP address on which the client will listen for the data connection to be opened. The syntax of this command is as follows:

```
PORT h1,h2,h3,h4,p1,p2
```

In this syntax, **h1** through **h4** are the four octets of an IP version 4 (IPv4) address. **p1** represents a number to be multiplied by 256, and then added to **p2** to obtain the port number. For example, assume that an FTP client issued the following PORT command to an FTP server:

```
PORT 10,1,200,201,9,8
```

This command instructs the FTP server to open a connection to IP address 10.1.200.201. It also tells the server that it should initiate the connection to port (9 x 256) + 8 = 2312.

By this definition of the PORT command's syntax, it becomes clear that the command was designed only to be suitable for an IPv4 address, because there are too few fields to accommodate an IPv6 address. A similar problem occurs when using the PASV command. Though the PASV command is issued with no arguments, the FTP server's RFC architected reply is as follows:

```
227 Entering Passive Mode. (h1,h2,h3,h4,p1,p2)
```

Again, **h1** through **h4** denote the IP address to which the client must initiate a connection, and (**p1** x 256) + **p2** denotes the port on which the server is listening. Therefore, the same IPv4 restrictions that exist for the PORT command also exist for PASV processing.

To overcome this limitation, RFC 2428 was created to replace the PORT and PASV commands with EPRT and EPSV, respectively. The EPRT command is processed similar to the PORT command, but the format is now as follows:

```
EPRT <d>protocol<d>address<d>port<d>
```

<d> Denotes a delimiter from the ASCII range of 33 to 126, though the ASCII 124 character (|) is recommended.

protocol Is the address family by which the client requests a connection. A value of **1** is a request for IPv4, and a value of **2** is a request for IPv6.

address Is the corresponding IPv4 or IPv6 address to which the connection must be opened. Port is simply the port to which the connection must be opened.

Two examples of the EPRT command in use are as follows (note that the IPv6 is simply the IPv4 10.1.200.201 address converted to IPv6 notation):

- IPv4: **EPRT |1|10.1.200.201|2312|**
- IPv6: **EPRT |2|::0A01:C8C9|2312|**

Similar to PASV, EPSV can be issued without an argument. The server's reply to the EPSV command has been designed as follows:

```
229 <text> (<d><d><d>port<d>)
```

The text is optional, and simply indicates to users that the passive mode will be implemented. The fields between the delimiters must be blank, because the EPSV command assumes that the FTP client will initiate a data connection to the

same IP address on which the control connection is established. An example of a possible server reply to an EPSV command is:

```
229 Entering Passive Mode (|||2312|)
```

Upon receiving this, the FTP client opens a data connection to port 2312 on the same IP address to which the control connection is active. This also useful when establishing a secure FTP session (see 14.1.9, “Securing FTP sessions” on page 527) across a firewall that implements Network Address Translation (NAT). NAT implementation during an FTP transfer requires the firewall to translate the IP address specified on the client’s PORT command or server’s PASV response. This allows local IP addresses to be routable from outside of the client or server’s subnet. However, if the control connection has been encrypted, the NAT implementation can no longer access these addresses, and the FTP applications might not be able to establish a data connection. However, by using the EPSV command, the FTP server already knows what IP address to use, and the data connection establishment can proceed without the intervention of the NAT implementation.

14.1.9 Securing FTP sessions

Although FTP provides security by requiring that users log on with a platform-specific user ID and password, this only prevents unauthorized access to the system itself. When transferring data from one host to another, the data within the packets (both on the control connection and the data connection) is sent in clear text. Therefore, network tools such as packet traces and sniffer devices can capture the packets and gain access to the transferred data. Additionally, the user ID and password used to log on to the server can be captured in these traces, giving a malicious user access to the system.

To avoid this problem, the design of FTP has been enhanced to make use of Transport Layer Security (TLS). TLS is defined in RFC 4366, and defines a standard of data encryption between two hosts. As denoted by its name, TLS is implemented on the transport layer, and thus applications using TLS do not need to know the specifics of RFC 4366. Instead, such applications only need to know how to invoke TLS, and in the case of FTP, this process is defined by RFCs 2228 and 4217.

The configuration and implementation of TLS varies by platform. However, RFCs 2228 and 4217 add to the FTP architecture the following TLS-related commands, which invoke TLS regardless of the platform’s specific configuration process:

AUTH Specifies the authentication method to be used in securing the FTP connection.

ADAT Passes Base64-encoded security data, specific to the mechanism specified on the AUTH command, used to in the security negotiation between the client and server.

PBSZ Specifies the largest buffer size in which encrypted data can be passed between the client and server.

PROT Specifies the data channel protection level. The argument passed must be one of the following:

- **C** Clear: Neither authentication nor encryption is used.
- **S** Safe: Authentication is performed, but no encryption is implemented.
- **E** Confidential: Encryption is performed, but no authentication is implemented.
- **P** Private: Both encryption and authentication are performed.

MIC Provides a Base64-encoded message used to integrity protect commands sent on the control connection.

CONF Provides a Base64-encoded message used to confidentially protect commands sent on the control connection.

ENC Provides a Base64-encoded message used to privately protect commands sent on the control connection.

CCC Instructs the FTP server that integrity-protected commands are no longer required for the FTP session.

Archived

Figure 14-6 shows an example of a common security negotiation using these commands.

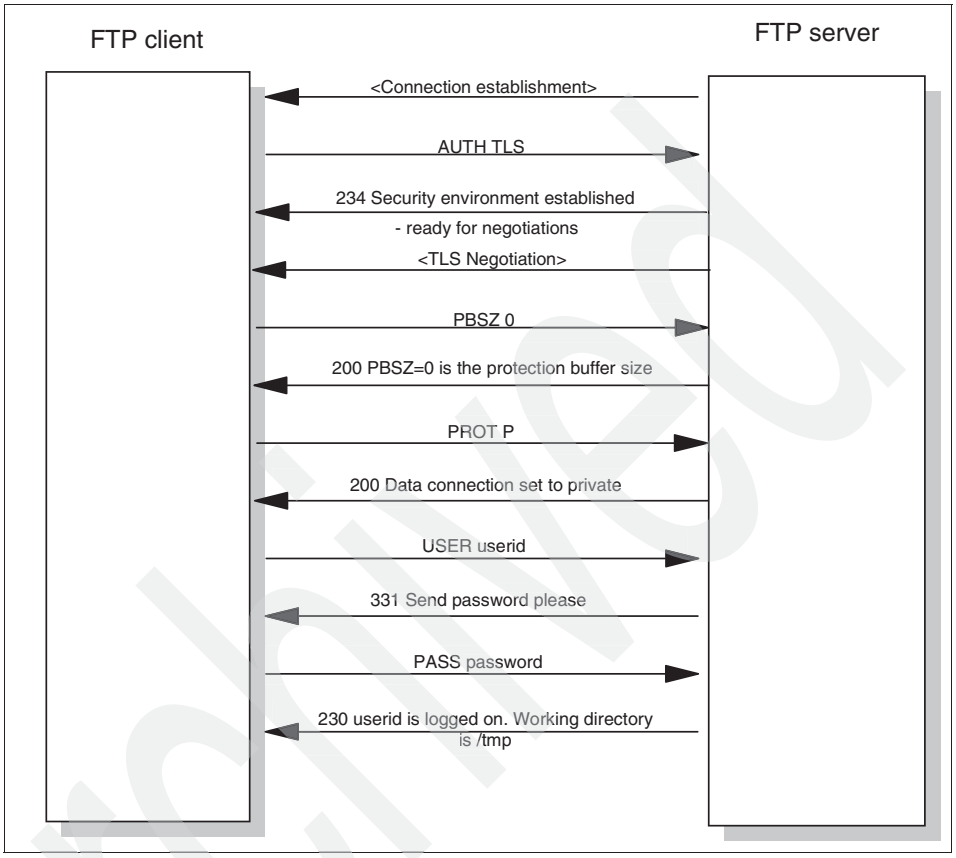


Figure 14-6 An example of FTP TLS processing

Most FTP client and server implementations provide configuration options to automate the security negotiation using the previous commands. This makes the implementation of secure FTP easier for the user because users do not need to understand or implement the details of the security during the FTP session itself, and only are prompted for a user ID and possibly a password.

14.2 Trivial File Transfer Protocol (TFTP)

The Trivial File Transfer Protocol (TFTP) is a standard protocol with STD number 33. Its status is elective and it is described in RFC 1350™ – The TFTP Protocol (Revision 2). Updates to TFTP are in the RFCs: 1785, 2347, 2348, and 2349.

TFTP file transfer is a disk-to-disk data transfer, and is an simple protocol used to transfer files. The simplicity of the architecture is deliberate in order to facilitate ease of implementation. This simplistic approach has many benefits over traditional FTP, including:

- Use by diskless devices to download firmware at boot time
- Use by any automated process for which the assignment of a user ID or password is not feasible
- Small application size, allowing it to be implemented inexpensively and in environments where resources are constricted

TFTP is implemented on top of the User Datagram Protocol (UDP, see 4.2, “User Datagram Protocol (UDP)” on page 146). The TFTP client initially sends read/write request through well-known port 69. The server and the client then determine the port that they will use for the rest of the connection. TFTP lacks most of the features of FTP (see 14.1, “File Transfer Protocol (FTP)” on page 514), and instead is limited to only reading a file from a server or writing a file to a server.

Note: TFTP has no provisions for user authentication; in that respect, it is an insecure protocol.

14.2.1 TFTP usage

The commands used by TFTP implementations are not architected by an RFC. Instead, only the direct interaction between a TFTP server and client are defined. Therefore, the commands used to invoke this interaction vary between different implementations of this protocol. However, each implementation has some variation of the following commands:

Connect <host>	Specifies the destination host ID.
Mode <ascii|binary>	Specifies the type of transfer mode.
Get <remote filename> [<local filename>]	Retrieves a file.
Put <remote filename> [<local filename>]	Stores a file.
Verbose	Toggles verbose mode, which displays additional information during file transfer, on or off.
Quit	Exits TFTP.

For a full list of these commands, see the user's guide of your particular TFTP implementation.

14.2.2 Protocol description

Every TFTP transfer begins with a request to read or write a file. If the server grants the request, the connection is opened and the file is sent in blocks of 512 bytes (fixed length). Blocks of the file are numbered consecutively, starting at 1, and each packet carries exactly one block of data.

Each data packet must be answered by an acknowledgment packet before the next one can be sent. Termination of the transfer is assumed on a data packet of less than 512 bytes. Although almost all errors will cause termination of the connection due to lack of reliability, TFTP can recover from packet loss. If a packet is lost in the network, a timeout occurs, initiating a retransmission of the last packet. This retransmission occurs both for lost data blocks or lost acknowledgements.

The requirement that every packet be acknowledged—including retransmissions—uncovered a design flaw in TFTP known as the Sorcerer's Apprentice Syndrome (SAS), described in RFC 783. On networks that experience latency or other delays, this flaw might cause excessive retransmission by both sides of the TFTP implementation. SAS was further documented in RFC 1123 and later corrected by adding OACK packets as a TFTP extension, described in RFC 2347. For additional details, refer to the appropriate RFCs.

14.2.3 TFTP packets

TFTP uses six types of packets, described in Table 14-3.

Table 14-3 TFTP packet types

Opcode	Operation
1	Read Request (RRQ)
2	Write Request (WRQ)
3	Data (DATA)
4	Acknowledgement (ACK)
5	Error (ERROR)
6	Option Acknowledgement (OACK)

The TFTP header contains the opcode associated with the packet. See Figure 14-7 for more details.

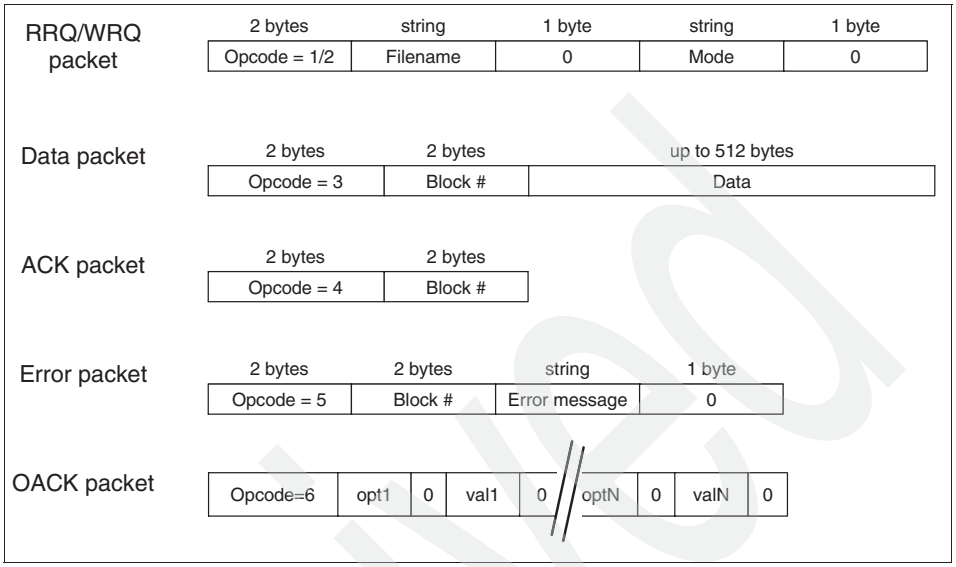


Figure 14-7 TFTP packet headers

14.2.4 Data modes

Two modes of transfer are currently defined in RFC 1350:

NetASCII US-ASCII, as defined in the USA Standard Code for Information Interchange with modifications specified in RFC 854 – Telnet Protocol Specification and extended to use the high order bit. (That is, it is an 8-bit character set, unlike US-ASCII, which is 7-bit.)

Octet Raw 8-bit bytes, also called binary.

The mode used is indicated in the TFTP header for the Request for Read/Write packets (RRQ/WRQ).

14.2.5 TFTP multicast option

The TFTP multicast option enables multiple clients to get files simultaneously from the server using the multicast packets. For example, when two similar machines are remotely booted, they can retrieve the same configuration file simultaneously by adding the multicast option to the TFTP option set. The TFTP

multicast option is described in RFC 2090. Figure 14-8 is an example of a TFTP read request packet modified to include the multicast option.



Figure 14-8 The TFTP Multicast header

If the server accepts the multicast, it sends an option acknowledgment (OACK) packet to the server including the multicast option. This packet consists of the multicast address and a flag that specifies whether the client should send acknowledgments (ACK).

14.2.6 Security issues

Because TFTP does not have any authentication mechanism, the server is responsible for protecting the host files. Generally, TFTP servers do not allow write access and only allow read access to public directories. Some server implementations also might employ host access lists to restrict access to only a subset of hosts.

14.3 Secure Copy Protocol (SCP) and SSH FTP (SFTP)

Secure Copy Protocol (SCP) and Secure Shell File Transfer Protocol (SFTP) are two methods of implementing secure transfer of files from one host to another. Neither protocol has been architected to perform authentication, and instead rely on the underlying SSH protocol on which they were built. This not only simplifies the implementation of the protocol, but avoids the pitfalls of traditional FTP (see 14.1, “File Transfer Protocol (FTP)” on page 514) by encrypting all data that passes between the two hosts.

Neither protocol is RFC architected, though an Internet draft currently exists for SFTP. Despite this lack of standardization, the protocol is implemented widely enough to be considered a de facto industry standard, and thus warrants discussion.

14.3.1 SCP syntax and usage

SCP functions much like the copy (**cp**) command on UNIX-based systems, and takes the following format:

```
scp flags sourceFile destinationFile
```

Both the user ID needed to gain access to the remote host and the location of the remote host are specified as part of the file name. For example, assume a user wants to copy a file from the host remote.host. The file is in the /tmp directory and is named yourdata. Additionally, this user has a user ID on the host of myId, and wants to copy the file to the /localfolder directory as mydata. In order to achieve this using SCP, the user issues the following command:

```
scp myId@remote.host:/tmp/yourdata /localfolder/mydata
```

Similarly, this user can upload a copy of mydata to remote.host, using the following command:

```
scp /localfolder/mydata myId@remote.host:/tmp/yourdata
```

Many implementations of SCP also include additional flags that allow further configuration of the SCP command. Although currently there are not RFC architected standards, the open source community has established the following flags for use with the protocol:

-c cipher	Selects the cipher to use for encrypting the data transfer.
-i identity_file	Selects the file from which the identity (private key) for RSA authentication is read.
-p	Preserves modification times, access times, and modes from the original file.
-r	Recursively copies entire directories.
-v	Verbose mode. Causes **scp** and **ssh(1)** to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.
-B	Selects batch mode.
-q	Disables the progress meter.
-C	Passes the **-C** flag to **ssh(1)** to enable compression.
-F ssh_config	Specifies an alternative per-user configuration file for SSH.
-P port	Specifies the port to connect to on the remote host.
-S program	Specifies the name of program to use for the encrypted connection. The program must understand **ssh(1)** options.
-o ssh_option	Can be used to pass options to SSH in the format used in **ssh_config(5)**. This is useful for specifying options for which there is no separate **scp** command-line flag. For example, forcing the use of protocol version 1 is specified using **scp -oProtocol=1**.
-4	Forces **scp** to use IPv4 addresses only.
-6	Forces **scp** to use IPv6 addresses only.

14.3.2 SFTP syntax and usage

SFTP allows a greater range of action than does SCP, but is still founded on the same principle of using SSH to perform secure transfers of data between systems. Unlike SCP, which only provides the capability of transferring a file, SFTP offers many of the same features found in traditional FTP. For this reason, a common mistake is to assume that SFTP is the RFC-architected FTP implemented using SSH. This is not the case, because FTP has a separate method of establishing data security (see 14.1.9, “Securing FTP sessions” on page 527). Instead, SFTP was designed independently of FTP.

Some of the additional functionality provided by SFTP over SCP include the ability to browse and manipulate the directory structure, rename files, and alter file permissions. There are three ways to invoke SFTP:

- **`sftp flags host`**

  This method of invoking SFTP enables a user to log on to the remote host. If SSH cannot establish the authenticity of the remote host, the user might be prompted for a password.

- **`sftp flags user@host:file`**

  This method of invoking SFTP allows SFTP to automatically retrieve the specified file.

- **`sftp flags user@host:directory`**

  This method of invoking SFTP logs the user onto the remote host and immediately places the user in the specified directory.

Similar to SCP, SFTP has a set of industry standard flags that can be configured when invoking SFTP:

`-b batchfile` Batch mode allows the SFTP commands to be entered into a file instead of requiring a user to enter each command sequentially. Because this mode does not allow user interaction, non-interactive authentication must be used. Additionally, in batch mode, SFTP aborts if any of the following commands experience a failure: **`get`**, **`put`**, **`rename`**, **`ln`**, **`rm`**, **`mkdir`**, **`chdir`**, **`lchdir`**, and **`lmkdir`**.

`-o ssh_option` Can be used to pass options to SSH.

`-s subsystem | sftp_server` Specifies the SSH2 subsystem or the path for an SFTP server on the remote host.

`-v` Raises the logging level. This option is also passed to SSH.

-B buffer_size	Specifies the size of the buffer that SFTP uses when transferring files.
-C	Enables compression.
-F ssh_config	Specifies an alternative per-user configuration file for SSH.
-P sftp_server_path	Connects directly to a local SFTP server.
-R num_requests	Specifies how many requests can be outstanding at any one time. Increasing this might slightly improve file transfer speed but will increase memory usage. The default is 16 outstanding requests.
-S program	Specifies the name of the program to use for the encrypted connection.
-1	Specifies the use of protocol version 1.

14.3.3 SFTP interactive commands

Many of the interactive commands offered by SFTP closely resemble those offered by a traditional FTP client (see 14.1.2, “FTP operations” on page 515). Indeed, the functionality afforded by both protocols is in many ways equivalent.

Directory navigation and manipulation

The following commands are used to not only navigate through a directory structure, but also alter paths within that structure. Note that fields enclosed in brackets [ ] are optional.

cd path	Changes remote directory to path.
lcd path	Changes local directory to path.
lmkdir path	Creates the local directory specified by path.
mkdir path	Creates the remote directory specified by path.
rmdir path	Removes the remote directory specified by path.

Archived

File manipulation

The following commands provide the ability to not only transfer files, but also to delete and rename files:

get [-P] remote-path [local-path]
Retrieves the file specified by remote-path and stores it on the local machine. If the local-path name is not specified, it is given the same name it has on the remote machine. If the **-P** flag is specified, the file's full permission and access time are copied, too.

put [-P] local-path [local-path]
Uploads the file specified by local-path and stores it on the remote host using the name specified by remote-path. If the remote-path name is not specified, it is given the same name it has on the local machine. If the -P flag is specified, the file's full permission and access time are copied, too.

rename oldpath newpath
Renames the remote file from oldpath to newpath.

rm path
Deletes the remote file specified by path.

Obtaining information

Use the following commands to obtain information in the SFTP session:

help
Displays help text.

lls [path]
Displays the local directory listing of either the specified path, or current directory if path is not specified.

lpwd
Prints the local working directory.

ls [-l] [path]
Displays the remote directory listing of either the specified path, or the current directory if path is not specified. If the **-l** flag is specified, it displays additional details including permissions and ownership information.

pwd
Displays remote working directory.

?
Synonym for help.

Shell manipulation

Because SFTP is built on the secure shell environment, users might need to obtain information about the shell, or issue shell commands without terminating SFTP. For this, use the following commands:

chgrp grp path
Changes the group of the file specified by path to grp. grp must be a numeric GID.

chmod mode path	Changes the permissions of the file specified by path to mode.
chown own path	Changes the owner of the file specified by path to own. own must be a numeric UID.
ln oldpath newpath	Creates a symbolic link from oldpath to newpath.
lumask umask	Sets the local umask to umask.
symlink oldpath newpath	Creates a symbolic link from oldpath to newpath.
! command	Executes the command in the local shell.
!	Escapes to the local shell.

Exiting SFTP

The following three commands terminate the SFTP session:

- **bye**
- **exit**
- **quit**

14.4 Network File System (NFS)

Designed by Sun Microsystems, the Network File System (NFS) protocol enables machines to share file systems across a network. The NFS protocol is designed to be machine-independent, operating system-independent, and transport protocol-independent. This is achieved through implementation on top of Remote Procedure Call (see 11.2.2, “Remote Procedure Call (RPC)” on page 415), which establishes machine independence by using the External Data Representation (XDR) convention.

Sun NFS is a proposed standard protocol with an elective status. The current NFS specifications are in RFC 1813 – NFS: NFS Version 3 Protocol Specification and RFC 3530 – NFS version 4 Protocol.

14.4.1 NFS concept

The NFS model allows authorized users to access files located on remote systems as though they were local. Two protocols in the model serve this purpose:

- The Mount protocol specifies the remote host and file system to be accessed and where to locate them in the local file hierarchy.

- The NFS protocol performs the file I/O to the remote file system.

Both the Mount and NFS protocols are RPC applications that implement the client/server model (see Figure 11-1 on page 409) and are transported by both TCP and UDP.

Mount protocol

The Mount protocol is an RPC application shipped with NFS and uses program number 100005. The MOUNT command acts as an RPC server program and provides a total of six procedures when accessing remote systems:

NULL	Does nothing. Useful for server response testing.
MNT	MOUNT function. Returns a file handle pointing to the directory.
DUMP	Returns the list of all mounted file systems.
UMNT	Removes a mount list entry.
UMNTALL	Removes all mount list entries for this client.
EXPORT	Returns information about the available file systems.

A file handle is a variable-length array with a maximum length of 64 bytes and is used by clients to access remote files. They are a fundamental part of NFS, because each directory and file are referenced only through file handles. For this reason, some implementations increase the security of the protocol by encrypting the handles. The file handles are obtained by executing the MOUNT application's MNT procedure, which locates the remote file system within the file hierarchy, and returns the file handle. Following the MOUNT command, a user can access the remote file system as though it were a part of the local file system.

For example, consider two remote hosts:

- HostA implements a hierarchical file system consisting of numerous directories and subdirectories.
- HostB does not implement a hierarchical file system, but instead implements a series of minidisks, with each minidisk appearing as a separate folder.

A user on HostA issues the MOUNT command to locally mount a minidisk from HostB. However, the user does not see the mounted volume as a minidisk, but instead sees it in the context of the hierarchical file system of HostA.

Although specific implementations can provide additional features, the generic syntax of the MOUNT command is as follows:

```
MOUNT -o options host:rvolume lvolume
```

Where:

options System-specific options, such as message size.

host The TCP/IP name of the remote host.

rvolume The remote volume to be mounted.

lvolume The location to which the remote volume will be mounted. This is also called the *mount point*.

In the case of our HostA and HostB example, assume the user wants to mount minidisk1 on HostB as /usr/lpp/tmp/md1. The user might issue the following command:

```
MOUNT HostB:minidisk1 /usr/lpp/tmp/md1
```

The user can access minidisk1 by simply navigating on the local file system to /usr/lpp/tmp/md1. This is illustrated in Figure 14-9.

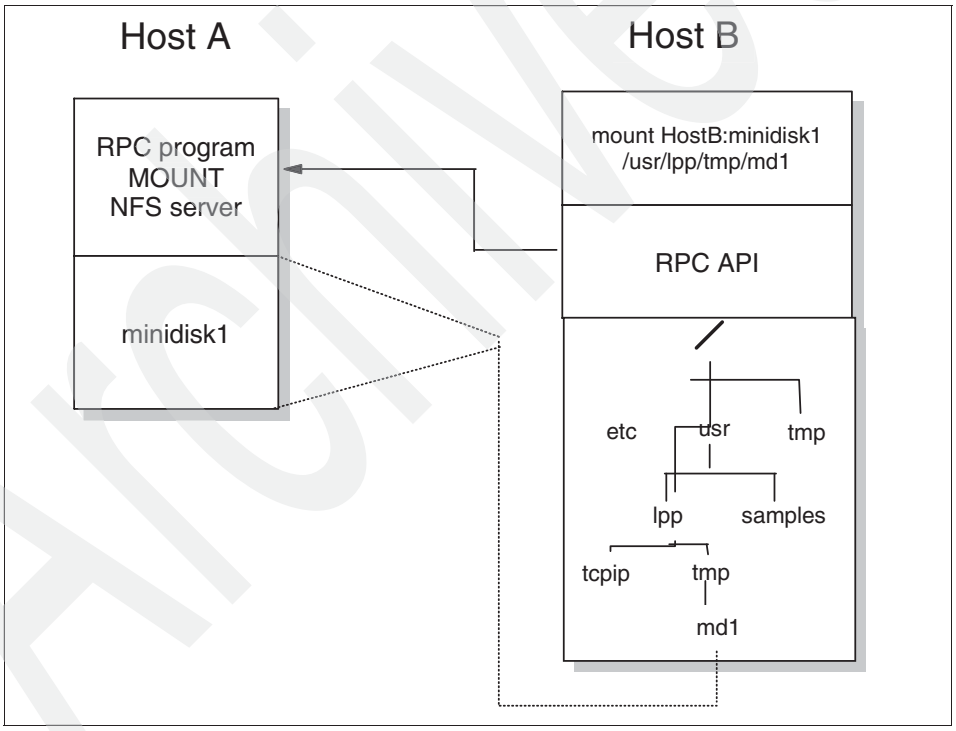


Figure 14-9 The NFS model

Archived

To better understand the MOUNT command, the syntax can be divided into three parts:

-o options The client portion. It is intended to be understood by the NFS client only. Therefore, valid options are client implementation-specific, and additional information must be sought from each implementation's documentation.

host:rvolume The server portion. The syntax of this argument depends on the server's file system. In our example, we used a generic HostB:minidisk1 argument. However, modify the argument to conform with the file system of the remote server. Additionally, the remote server can include additional parameters within this argument, such as user authentication or file system attributes. Refer to the documentation of the specific NFS server implementation to determine what parameters it will accept.

lvolume A client part. This is called the mount point. This is where the remote file system will be hooked into the local file system.

The UMOUNT command removes the remote file system from the local file hierarchy. Continuing the previous example, the following command removes the /usr/lpp/tmp/md1 directory:

```
UMOUNT /usr/lpp/tmp/md1
```

NFS protocol

NFS is the RPC application program providing file I/O functions to a remote host after it has been requested through a MOUNT command. It has program number 100003 and sometimes uses IP port 2049. Because this is not an officially assigned port and several versions of NFS (and Mount) already exist, port numbers can change. We advise you to use portmap or RPCBIND (see 11.2.2, "Remote Procedure Call (RPC)" on page 415) to obtain the port numbers for both the Mount and NFS protocols. The NFS protocol is transported by both TCP and UDP.

The NFS program supports 22 procedures, providing for all basic I/O operations, including:

ACCESS Resolves the access rights, according to the set of permissions of the file for that user.

LOOKUP Searches for a file in the current directory and if found, returns a file handle pointing to it plus information about the file's attributes.

READ/WRITE Basic read/write primitives to access the file.

RENAME Renames a file.

REMOVE	Deletes a file.
MKDIR, RMDIR	Creates and deletes subdirectories.
GET, SET-ATTR	Gets or sets file attributes.

Other functions are also provided.

These correspond to most of the file I/O primitives used in the local operating system to access local files. In fact, after the remote directory is mounted, the local operating system has only to re-route the file I/O primitives to the remote host. This makes all file I/Os look alike, regardless of whether the file is located locally or remotely. The user can operate his or her normal commands and programs on both kinds of files; in other words, this NFS protocol is completely transparent to the user (see Figure 14-10).

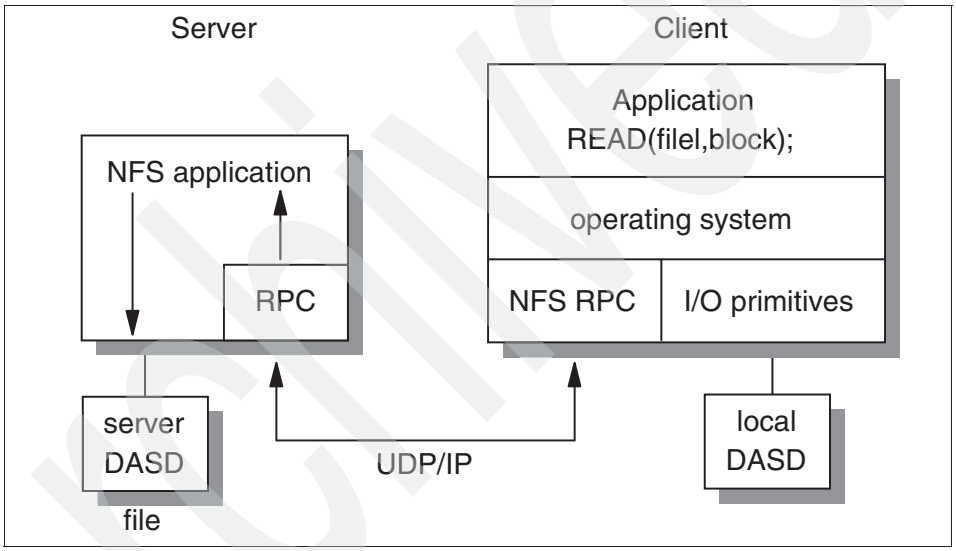


Figure 14-10 NFS: File I/O

14.4.2 File integrity

Because NFS is a stateless protocol, there is a need to protect the file integrity of the NFS-mounted files. Many implementations have the Lock Manager protocol for this purpose. Sometimes, multiple processes open the same file simultaneously. If the file is opened for merely read access, every process will get the most current data. If there is more than one processes writing to the file at one time, the changes made by writers should be coordinated, and at the same time, the most accurate changes should be given to the readers. If some part of the file is cached on each client's local system, it becomes a more complicated

job to synchronize the writers and readers accordingly. Though they might not occur so frequently, take these incidents into consideration.

14.4.3 Lock Manager protocol

This protocol allows client and server processes to exclude the other processes when they are writing to the file. When a process locks the file for exclusive access, no other process can access the file. When a process locks the file for shared access, the other processes can share the file but cannot initiate an exclusive access lock. If any other process request conflicts with the locking state, either the process waits until the lock is removed or it returns an error message.

14.4.4 NFS file system

NFS assumes a hierarchical, or directory-based, file system. In such a system, files are unstructured streams of uninterpreted bytes; that is, files are seen as a contiguous byte stream, without any record-level structure.

With NFS, all file operations are synchronous. This means that the file operation call only returns when the server has completed all work for this operation. In case of a write request, the server will physically write the data to disk and if necessary, update any directory structure before returning a response to the client. This ensures file integrity.

NFS also specifies that servers should be stateless. That is, a server does not need to maintain any extra information about any of its clients in order to function correctly. In case of a server failure, clients only have to retry a request until the server responds without having to reiterate a MOUNT operation.

14.4.5 NFS version 4

NFS in version 4 (NFSv4) is defined in RFC 3530. The NFS model and concepts are essentially unchanged from version 3 (RFC 1813), and the new version focuses on improving performance, increasing security, augmenting cross-platform interoperability, and providing a design for protocol extensions.

Removal of ancillary protocols

As noted previously, the NFS model incorporated numerous other protocols, such as the Mount protocol and the Lock Management protocol, in order to address gaps in the basic NFS protocol. The use of the Mount protocol in pervious versions of NFS was solely for the purpose of obtaining the initial file handle for individual servers. This has been replaced by the combination of

public file handles and the LOOKUP, allowing the NFSv4 protocol to mount file systems without invoking additional protocols. Additionally, NFSv4 integrates locking schemes into the base NFS protocol, and therefore no longer relies upon the Lock Manager protocol. These integrations are transparent to users.

Introduction of stateful operations

Previous versions of NFS were considered stateless. However, with NFSv4, OPEN and CLOSE operations were added, creating a concept of statefulness on the NFS server. This addition provides numerous advantages over the stateless model, including:

- The ability to integrate enhanced locking functionality
- The ability for a server to delegate authority to a client
- Allowing aggressive caching of file data

The total number of operations in NFSv4 has expanded to 38. Many of these perform functions similar to those provided in previous releases of NFS, but new operations have been added to integrating locking, security, and client access. Examples of some these include:

`lock`	Creates a lock on a file.
`lockt`	Tests for the existence of a lock on a file.
`lucku`	Unlocks a file.
`secinfo`	Retrieves NFS security information.

Security enhancements

In version 4, NFS now takes advantage of available security methods, including:

- Support for RPCSEC_GSS, described in RFC 2203
- Support for Kerberos 5, described in RFC 4120
- Support for the Low Infrastructure Public Key Mechanism (LIPKEY), defined in RFC 2847
- Support for the Simple Public-Key GSS-API Mechanism (SPKM-3), defined in RFC 2025

Additional changes

The following additional changes, which do not fit into the previous discussions of NFSv4, are worth noting:

- New volatile style file handles, which help server implementations cope with file system reorganizations

- Compound commands for performance, which execute **lookup** and **read** commands in one operation
- Server delegation of status and locks to a given client, improving caching and performance
- Internationalization in the form of 16/32-bit character support for file names by means of the UTF-8 encoding

14.4.6 Cache File System

The Cache File System (CacheFS™) provides the ability to cache one file system on another. CacheFS accomplishes caching by mounting remote directories on the local system. Whenever the client needs a mounted file, it first refers to local cache. If the requested file exists, it is accessed locally. Otherwise, the file is retrieved from the server using NFS. If reading the file sequentially, the future data is retrieved for future access to the cache, increasing the file access speed. Whether or not the cache is stored in random access memory (RAM) or disk is implementation specific. However, if the size the cache is not prohibitive, storing the cache on RAM is preferred because it provides much faster access.

CacheFS periodically checks the cached data for the data accuracy, sweeps out the inaccurate data from the cache, and retrieves the current data from the server to the cache. Although this reduces server and network loads, improving performance for clients on slow links, CacheFS is only proper for the cases in which the files are not changed frequently.

14.4.7 WebNFS

WebNFS™ is an enhanced version of the standard NFS. WebNFS enables clients to access files over wide area networks (WANs). Because, as a transport protocol, UDP is fast for local area networks (LANs), many implementations use UDP for the standard NFS. However, a TCP connection is much more reliable for wide area networks. Most of the new NFS implementations support TCP and UDP. WebNFS clients first attempt to use a TCP connection. If it fails or is refused, it then uses a UDP connection. WebNFS can easily recover from dropped lines and recover the lost data.

Path name evaluation

The standard NFS version 3 is designed for LANs. The amount of time for each LOOKUP request is not significant. As a result of this, the NFS protocol permits only one single path name request at a time. When we consider a WAN or the Internet, to request the path name and evaluate the path name might cause significant delays. This process is very expensive, especially for files that are

several directories away. WebNFS can handle retrieving the entire path name and evaluate it.

NFS URL

This scheme enables the NFS clients to go over the Internet, as do other Internet applications. The URL scheme is based on the common Internet scheme syntax, and is described in RFC 2224. The general form of the NFS URL scheme is:

```
nfs://<host>:<port><url-path>
```

The port is optional. If it is not specified, the default value is 2049. The WebNFS server evaluates the path using a multicomponent lookup relative to the public file handle.

14.5 The Andrew File System (AFS)

The Andrew File System (AFS) is a distributed file system used in non-Distributed Computing Environments (DCEs). The DCE Distributed File System (DFS) was based on AFS, and the two file systems are similar in architecture. AFS offers high availability by storing information about more than one server. It is also designed to work over WANs or the Internet, unlike NFS version 3.

The latest version (AFS 3) has the following attributes:

- Single logical shared namespace
  Every AFS user shares the same uniform name space. File names are independent of both the user's and the file's physical locations. Groups of client and server machines are known as cells.
- Client caching
  Data is cached on client machines to reduce subsequent data requests directed at file servers, which reduces network and server loads. Servers keep track of client caches through callbacks, guaranteeing cache consistency without constant queries to the server to see if the file has changed.
- RPCs
  AFS uses its Remote Procedure Call (RPC) reads and writes for efficient data transfer across the network.
- Security
  Kerberos-based authentication requires that users prove their identities before accessing network services. When authenticated, AFS access control lists (ACLs) give individual users or groups of users varying levels of authority to perform operations on the files in a directory.

- Replication
  Replication techniques are used for file system reliability. Multiple copies of applications and data can be replicated on multiple file servers within a cell. When accessing this information, a client chooses among the available servers. Replication also reduces the load on any particular server by placing frequently accessed information on multiple servers.
- Management utilities
  Backup, reconfiguration, and routine maintenance are all done without any system down time and files remain available to users during these operations. This is done by creating online clones of volumes (subsets of related files).

AFS commands are RPC-based. Administrative commands can be issued by any authenticated administrator from any client workstation. System databases track file locations, authentication information, and access control lists. These databases are replicated on multiple servers and are dynamically updated as information changes.

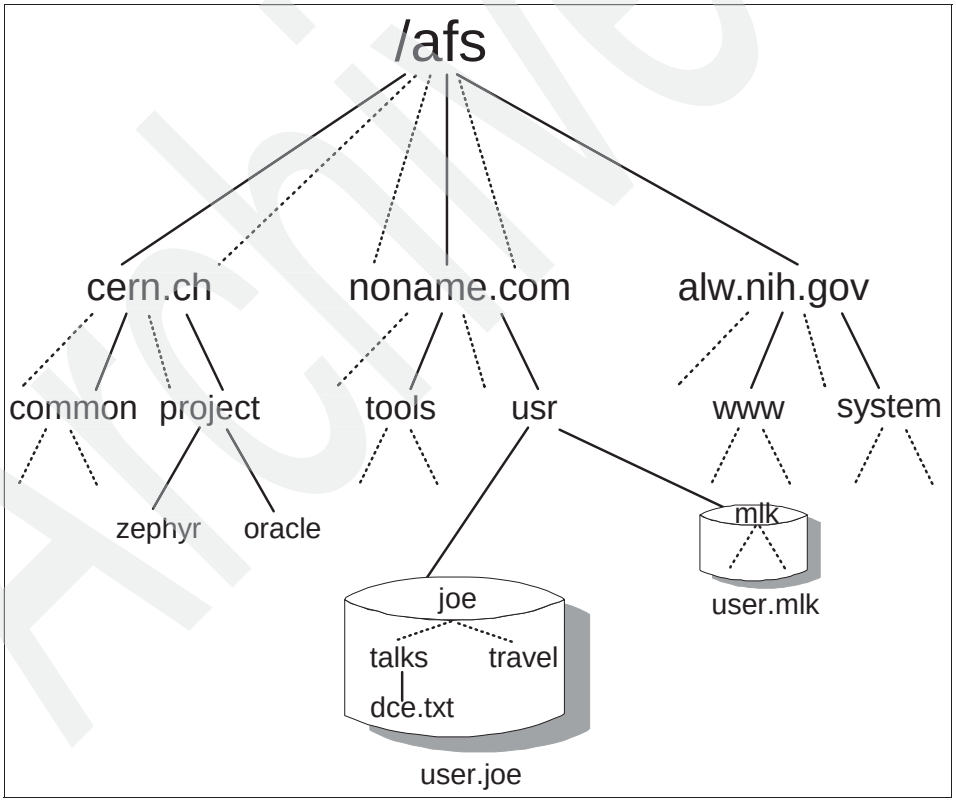


Figure 14-11 The AFS 3 namespace

Archived

Figure 14-11 on page 547 shows an example of an AFS namespace. user.joe is a volume, which is a group of files managed as a single unit. Any user, situated anywhere in the network, can access Joe's dce.txt file by using the file name:

/afs/noname.com/usr/joe/talks/dce.txt

For further information about AFS, refer to the OpenAFS Web site at:

http://www.openafs.org/

14.6 Common Internet File System (CIFS)

The Common Internet File System (CIFS) was developed by IBM in 1985 to be a protocol for sharing files, printers, and serial ports. Originally named Server Message Block (SMB), Microsoft began developing it in 1987. SMB/CIFS adheres to the client/server model of request and response (see 11.1.1, “The client/server model” on page 408). Although the original description of the protocol provides brief details about how SMB/CIFS can run on top of TCP/IP, currently the most common implementation is to run SMB/CIFS on top of NetBIOS over TCP/IP. This process is defined in RFCs 1001 and 1002, both of which comprise STD0019.

14.6.1 NetBIOS over TCP/IP

While the process of implementing NetBIOS over TCP/IP is beyond the scope of this tutorial, the functions provided by NetBIOS used by SMB/CIFS are not. In particular, three services provided by NetBIOS are essential to the SMB/CIFS protocol.

Name Service

Much like the method in which the Domain Name System (DNS, see Chapter 12, “Directory and naming protocols” on page 425) maps host names to IP addresses, NetBIOS provides a service to map human-readable computer names to IP addresses. However, while most DNS implementations require that an administrator somewhere manually adds mappings into a configuration, NetBIOS provides the ability for computers to dynamically register their name and IP address at startup. This is achieved by two different methods:

b-node name registration The client workstation builds a *name registration packet*, which is then broadcast on UDP port 137. Any other computers in the network that receive this packet but already have registered the name will then send back a reply, called a *defense packet* indicating that there is a conflict.

If the client workstation initiating the registry receives no defense packets, it is assumed that its name has been successfully registered.

p-node name registration This method occurs much in the same manner as b-node registration. However, instead of broadcasting the name registration packet on UDP port 137, the client workstation unicasts the packet directly to a NetBIOS Name Server (NBNS). If the requested name has already been registered with the NBNS, a negative reply is returned to the client. Otherwise, a positive reply indicates the acceptance of the name registration.

One other service offered by the Name Service is the name query. Much like DNS, this allows clients to query NetBIOS for the IP address of another computer in the network. To use this service, a client workstation builds a *name query request*, which is sent on UDP port 137. Again, there are two methods for this. The b-node method broadcasts the packet and waits for a response from the workstation that owns the name. In the p-node method, the workstation sends the request directly to a NBNS server.

Session Service

RFC 1001 indicates that NetBIOS should provide reliable sessions. It also indicates that message exchanges should be fully-duplexed and sequenced. Therefore, the Session Service offered by NetBIOS mimics the TCP architecture (see 4.3, “Transmission Control Protocol (TCP)” on page 149). The Session Service uses TCP port 139 to achieve this and includes the following network primitives, all of which can be compared to socket API calls (see 11.2.1, “The socket API” on page 410):

`call` Creates a NetBIOS session. This is similar to the **`connect()`** socket API call.

`listen` Awaits a **`call`** command from another host. This is similar to the **`listen()`** socket API call.

`send` Sends a message over a NetBIOS session. This is similar to the **`sendmsg()`** socket API call.

`receive` Receives a message over a NetBIOS session. This is similar to the **`recvmsg()`** socket API call.

`hang up` Terminates a session. This is similar to the **`close()`** socket API call.

Datagram Service

Just as the Session Service mimics the functionality of the TCP architecture, the Datagram Service mimics the functionality of UDP architecture (see 4.2, “User Datagram Protocol (UDP)” on page 146). RFC 1001 indicates that this service should be unreliable, non-sequenced, and connectionless. This is achieved using UDP port 138.

14.6.2 SMB/CIFS specifics

As discussed previously, SMB/CIFS follows the client/server model. It is command based, each command corresponding to a request, followed by a reply from the remote host. These commands allow SMB clients to perform things such as user logons, obtain a lock, read from or write to a file, and terminate a session.

A quick note about locks

SMB/CIFS implements opportunistic locks, or *oplocks*, which enable clients to access files on a remote host without compromising file integrity. These locks are as follows:

Exclusive oplocks This lock opens the file for reading and writing, and prevents other clients from accessing the file. This allows a client to improve network performance by reading ahead and buffering data to be written. Reading ahead simply means that a client can transfer more data than has been requested by the user without fear of the data having changed by the time the user actually requests the data. Buffering data to be written means that a user can enter data to be changed, but instead of sending every change to the server, the client sends a block of changes.

Batch oplocks This lock allows the client to lock a file for reading and writing and keeps the file open even though the user might have closed the file. This improves performance for applications that might be opening and closing the same file many times.

Level II oplocks This lock allows the file to opened for reading. This allows multiple clients to have a single file open, but does not allow any of those clients to write to the file.

Archived

Dialects

Another characteristic of SMB/CIFS is that there are many variations, called dialects, of the protocol. Therefore, when first establishing a session, the two hosts must negotiate the dialect they will use throughout the remainder of the session. It is also for this reason that there are very many SMB/CIFS commands (we do not discuss all the commands here). We provide Figure 14-12 on page 552 as an example of a typical SMB/CIFS exchange to write to a file. The illustrated SMB commands are written in generic terms, which SMB implements as follows:

1. Establishing a session is accomplished not through SMB/CIFS commands, but rather through the NetBIOS implementation alone. SMB/CIFS uses the NetBIOS session service, which then creates the TCP connection with the remote host. However, depending on the amount of information provided to NetBIOS by SMB/CIFS, this might be preceded by a name query to determine the IP address of the remote host.
2. The SMB_COM_NEGOTIATE message negotiates a dialect. This includes a list of dialects supported by the SMB/CIFS client. The server, receiving the message, selects from this list a dialect it supports and sends this back in the reply.
3. The user login is performed through the SMB_COMM_SESSION_SETUP_ANDX command. This transmits the user's name and credentials to the server for verification.
4. Opening a file requires two steps:
   a. First, the client sends an SMB_COM_TREE_CONNECT_ANDX message, which tells the SMB/CIFS server the name of the shared disk desired by the client.
   b. Next, the client sends an SMB_COM_OPEN_ANDX message containing the name of the file within that disk to which the client wants access. At this time, the SMB/CIFS server grants the appropriate oplock to the client for the requested file. Because this is an example of a sequence to write to a file, an exclusive oplock is granted to the client.
5. Writing to the file is done through the SMB_COM_WRITE command.
6. When the client has finished updating the file, it closes the file using the SMB_COM_CLOSE command.
7. To disconnect, the client must first disconnect from the disk, which is done using the SMB_COM_TREE_DISCONNECT message. After the server returns a successful reply, the client can instruct NetBIOS to close the session by sending the **`hang up`** command to the remote host's NetBIOS.

Archived

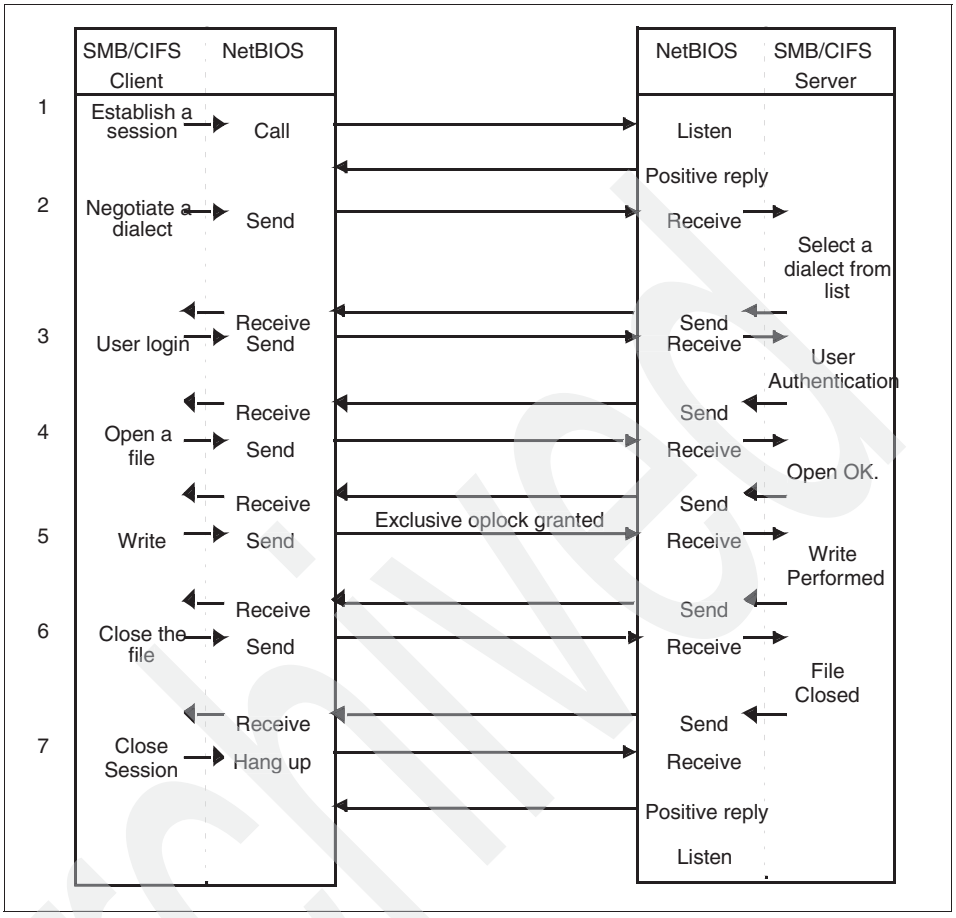


Figure 14-12 SMB/CIFS on NetBIOS over TCP/IP model

14.7 RFCs relevant to this chapter

The following RFCs provide detailed information about the application protocols and architectures presented throughout this chapter:

- RFC 0854 – Telnet Protocol Specification (May 1983)
- RFC 0959 – File Transfer Protocol (FTP) (October 1985)
- RFC 1001 – Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods (March 1987)
- RFC 1002 – Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications (March 1987)

- RFC 1123 – Requirements for Internet Hosts -- Application and Support (October 1989)
- RFC 1350 – The TFTP Protocol (Revision 2) (July 1992)
- RFC 1579 – Firewall-Friendly FTP (February 1994)
- RFC 1785 – TFTP Option Negotiation Analysis (March 1995)
- RFC 1813 – NFS Version 3 Protocol Specification (June 1995)
- RFC 2054 – WebNFS Client Specification (October 1996)
- RFC 2055 – WebNFS Server Specification (October 1996)
- RFC 2090 – TFTP Multicast Option (February 1997)
- RFC 2203 – RPSEC_GSS Protocol Specification (September 1997)
- RFC 2224 – NFS URL Scheme (October 1997)
- RFC 2228 – FTP Security Extensions (October 1997)
- RFC 2347 – TFTP Option Extension (May 1998)
- RFC 2348 – TFTP Blocksize Option (May 1998)
- RFC 2349 – TFTP Timeout Interval and Transfer Size Options (May 1998)
- RFC 2428 – FTP Extensions for IPv6 and NATs (September 1998)
- RFC 2577 – FTP Security Considerations (May 1999)
- RFC 3530 – Network File System (NFS) version 4 Protocol (April 2003)
- RFC 4120 – The Kerberos Network Authentication Service (V5) (July 2005)
- RFC 4217 – Securing FTP with TLS (October 2005)

Archived

Archived

15

Mail applications

Electronic mail (e-mail) is probably the most widely used TCP/IP application in the Internet community today. For most people, it has become an integral part of everyday life, and it serves as the primary communication vehicle for most businesses. This chapter provides an overview of the TCP/IP application protocols dealing with electronic mail, including:

- Simple Mail Transport Protocol (SMTP)
- Sendmail
- Multipurpose Internet Mail Extensions (MIME)
- Post Office Protocol (POP)
- Internet Message Access Protocol (IMAP)

© Copyright IBM Corp. 1989-2006. All rights reserved.

15.1 Simple Mail Transfer Protocol

The basic Internet mail protocols provide mail (note) and message exchange between TCP/IP hosts, but generally require that data be represented as 7-bit ASCII text. Because this can be restrictive, facilities have been added for the transmission of data that cannot be represented in this manner. Originally, there were three standard protocols that apply to mail of this kind. The term Simple Mail Transfer Protocol (SMTP) is frequently used to refer to the combined set of protocols because they are so closely interrelated. Strictly speaking, however, SMTP is just one of the three. Those three standards were:

- A standard for the exchange of mail between two computers (STD 10/RFC 821), which specified the protocol used to send mail between TCP/IP hosts. This standard was SMTP itself.
- A standard (STD 11) on the format of the mail messages, contained in two RFCs. RFC 822 described the syntax of mail header fields and defined a set of header fields and their interpretation. RFC 1049 described how a set of document types other than plain text ASCII can be used in the mail body (the documents are 7-bit ASCII containing imbedded formatting information: PostScript, Scribe, SGML, TEX, TROFF, and DVI are all listed in the standard).

  The official protocol name for this standard was MAIL.
- A standard for the routing of mail using the Domain Name System, described in RFC 974. The official protocol name for this standard was DNS-MX.

However, because of the ambiguity of having three standards describing a common application, two new RFCs were released to consolidate, update, and clarify the basic protocol for Internet electronic mail transport:

- RFCs 821 and 974 were consolidated into RFC 2821, which is not the current STD 10. However, RFC 2821 notes that it does not change or add new functionality from that listed in 821 and 974. Instead, it simply serves to remove the ambiguity from the collection of standards. It also discusses in greater depth aspects from those RFCs that have proven to be important in recent implementations of SMTP.
- RFC 822 is similarly obsoleted by RFC 2822.

RFC 2821 dictates that data sent through SMTP is 7-bit ASCII data, with the high-order bit cleared to zero. This is adequate in most instances for the transmission of English text messages, but is inadequate for non-English text or non-textual data. There are two approaches to overcoming these limitations:

- Multipurpose Internet Mail Extensions (MIME), defined in RFCs 2045 to 2049, which specifies a mechanism for encoding text and binary data as 7-bit ASCII within the mail envelope defined by RFC 2822. MIME is described in 15.3, “Multipurpose Internet Mail Extensions (MIME)” on page 571.
- SMTP service extensions, which define a mechanism to extend the capabilities of SMTP beyond the limitations imposed by RFC 2821. In the context of overcoming the 7-bit ASCII limitation, the two most important extensions are defined in RFC 1652 and RFC 1870. However, there are multiple current RFCs that describe other SMTP service extensions, some of the more notable of which are detailed in Table 15-1.

Table 15-1 SMTP service extensions

RFC	Description
1652	A protocol for 8-bit text transmission that allows an SMTP server to indicate that it can accept data consisting of 8-bit bytes. A server that reports that this extension is available to a client must leave the high order bit of bytes received in an SMTP message unchanged if requested to do so by the client.
1845	A protocol to allow an interrupted SMTP transaction to be restarted at a later time. This prevents the need to repeat all of the commands, or retransmit data that had been successfully received prior to the interruption.
1846	This extension allows an Internet host implementing SMTP to return a 521 message, indicating that it does not accept incoming mail.
1870	A protocol for message size declaration that allows a server to inform a client of the maximum size message it can accept. Without this extension, a client can only be informed that a message has exceeded the maximum size acceptable to the server (either a fixed upper limit or a temporary limit imposed by a lack of available storage space at the server) after transmitting the entire message. When this happens, the server discards the failing message. If both the client and server support the message size declaration extension, the client can declare an estimated size of the message to be transferred and the server will return an error if the message is too large.
2034	This extension defines enhanced error codes to allow SMTP the ability to return more informative explanations of error conditions.
2554	This introduces an option to allow an SMTP client and server to negotiate an authentication mechanism, and optionally negotiate a security layer to protect subsequent protocol interactions.

RFC	Description
3030	This introduces two extensions. The first allows SMTP clients to use a BDAT command as an alternative to the DATA command (see Figure 15-2 on page 564). This provides the ability to efficiently send large MIME messages. The second extension takes advantage of the new BDAT command to permit sending of MIME messages that employ the binary transfer encoding.
3207	This introduces the ability for SMTP clients and servers to implement Transport Layer Security (TLS, see 22.8, “Transport Layer Security (TLS)” on page 861) in their protocol interactions, the sending of data, or both.
3461	This introduces the ability for SMTP clients to request notifications of delivery status, allowing clients to know if messages have been successfully delivered, delayed, and so on.
3885	This extends RFC 3461 to allow clients to track the sending of a message when needed replies described in 3461 are not received.

Note that the RFC 1652 and MIME approaches are complementary rather than competing standards. In particular, RFC 1652 is titled “SMTP Service Extension for 8-bit-MIMEtransport,” because the MIME standard allows messages to be declared as consisting of 8-bit data rather than 7-bit data. Such messages cannot be transmitted by SMTP agents that strictly conform to RFC 2821, but can be transmitted when both the client and the server conform to RFC 1652. Whenever a client SMTP attempts to send 8-bit data to a server that does not support this extension, the client SMTP must either encode the message contents into a 7-bit representation compliant with the MIME standard or return a permanent error to the user.

Additionally, the service extension defined in RFC 1652 does not permit the sending of arbitrary binary data, because RFC 2821 defines the maximum length of a line that an SMTP server is required to accept as 1000 characters. However, non-text data can easily have sequences of more than 1000 characters without a <CRLF> sequence.

Note: The service extension specifically limits the use of non-ASCII characters (those with values above decimal 127) to message bodies. They are *not* permitted in RFC 2822 message headers.

15.1.1 How SMTP works

SMTP is based on *end-to-end delivery*: An SMTP client contacts the destination host's SMTP server directly, on well-known port 25, to deliver the mail. It keeps the mail item being transmitted until it has been successfully copied to the recipient's SMTP. This is different from the store-and-forward principle that is common in many mailing systems, where the mail item can pass through a number of intermediate hosts in the same network on its way to the destination and where successful transmission from the sender only indicates that the mail item has reached the first intermediate hop.

In various implementations, it is possible to exchange mail between the TCP/IP SMTP mailing system and the locally used mailing systems. These applications are called *mail gateways* or *mail bridges*. Sending mail through a mail gateway can alter the end-to-end delivery specification, because SMTP only guarantees delivery to the mail-gateway host, not to the real destination host located beyond the TCP/IP network. When a mail gateway is used, the SMTP end-to-end transmission is host-to-gateway, gateway-to-host, or gateway-to-gateway; the behavior beyond the gateway is not defined by SMTP.

SMTP messages

In SMTP, each message has:

- A header, or envelope, the structure of which is strictly defined by RFC 2822

  The mail header is terminated by a null line (that is, a line with nothing preceding the <CRLF> sequence).

- Contents

  Everything after the null (or blank) line is the message body, which is a sequence of lines containing ASCII characters (that is, characters with a value less than 128 decimal).

RFC 2821 defines a client/server protocol (see 11.1.1, “The client/server model” on page 408). As usual, the client SMTP (referred to as the *sending* SMTP) is the entity that initiates the session, and the server (referred to as the *receiving* SMTP) is the one that responds to the session request. Because the client SMTP frequently can also act as a server for a user mailing program, it is often simpler to refer to the client as the sender SMTP and to the server as the receiver SMTP.

The SMTP destination address

Also known as the *mailbox address*, the general form of the destination address is *local-part@domain-name* and can take several forms:

user@host For a direct destination on the same TCP/IP network.

user%remote-host@gateway-host
For a user on a non-SMTP destination remote-host, through the mail gateway-host.

@host-a,@host-b:user@host-c
For a relayed message. This contains explicit routing information. The message is first delivered to host-a, who re-sends (relay) the message to host-b. Host-b then forwards the message to the real destination host-c. Note that the message is stored on each of the intermediate hosts, so we do not have an end-to-end delivery in this case.

Mail header format

Typically, a user does not have to worry about the message header because it is managed by SMTP itself. A short reference is included here for completeness.

RFC 2822 contains a complete lexical analysis of the mail header. The syntax is written in a form known as the Augmented Backus-Naur Form (ABNF), defined in RFC 2822. Additionally, many RFCs related to RFC 2822 use this format. RFC 2822 also describes how to parse a mail header to a *canonical representation*, unfolding continuation lines, deleting insignificant spaces, removing comments, and so on. The syntax is powerful, but relatively difficult to parse. A basic description is given here, which should be adequate for you to interpret the meaning of simple mail headers that might be encountered. However, this description is too great a simplification to understand the details workings of RFC 2822 mailers; for a full description, refer to RFC 2822.

Briefly, the header is a list of specifications in the form of:

```
keyword: value
```

Fields begin in column 1. Lines beginning with white space characters (space or tab) are continuation lines that are unfolded to create a single line for each field in the canonical representation. Strings enclosed in ASCII quotation marks indicate single tokens within which special characters, such as the colon, and are not significant. Many important field values (such as those for the *To* and *From* fields) are *mailboxes*. The most common forms for these are:

```
yourEmail@yourdiv.redbookscorp.com
Your Email <yourEmail@yourdiv.redbookscorp.com>
"Your Email" <yourEmail@yourdiv.redbookscorp.com>
```

In this example, the string *Your Email* is intended to be read by human recipients and is the name of the mailbox owner. *yourEmail@yourdiv.redbookscorp.com* is the machine-readable address of the mailbox (the angle brackets delimit the address but are not part of it), defined previously in “The SMTP destination

address” on page 559. You can see that this form of addressing is closely related to the Domain Name System concept (see 12.2, “Dynamic Domain Name System” on page 453). In fact, the client SMTP uses the Domain Name System to determine the IP address of the destination mailbox (see 15.1.2, “SMTP and the Domain Name System” on page 565).

Table 15-2 lists some frequently used fields.

Table 15-2 Common SMTP header fields

Keyword	Value
To	Primary recipients of the message.
cc	Secondary (carbon-copy) recipients of the message.
From	The sender of the message.
Reply-to	The mailbox to which responses are to be sent. This field is added by the sender.
Return-path	The addresses and route by which the sender can be reached. This field is added by the final transport system that delivers the mail.
Subject	A summary of the message being sent. This is specified by the sender.

A sample header might appear as follows:

```
From: myEmail@mydiv.redbookscorp.com
To: “Your Email” <yourEmail@yourdiv.redbookscorp.com>
cc: “Your Boss” <yourBoss@yourdiv.redbookscorp.com>
Reply-To: myEmail@mydiv.redbookscorp.com
Subject: This is a sample SMTP header
```

Mail exchange

The SMTP design is based on the model of communication shown in Figure 15-1 on page 562. As a result of a user mail request, the sender SMTP establishes a two-way connection with a receiver SMTP. The receiver SMTP can be either the ultimate destination or an intermediate (mail gateway). The sender SMTP will generate commands that are replied to by the receiver SMTP.

Archived

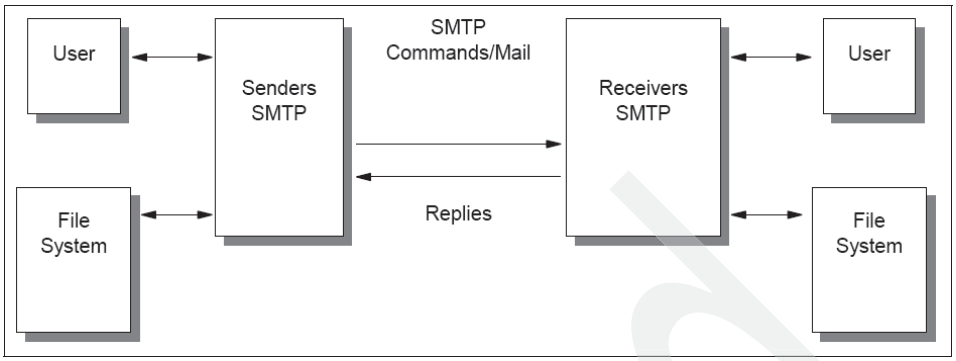


Figure 15-1 The SMTP model

SMTP mail transaction flow

Although mail commands and replies are rigidly defined, the exchange can easily be followed in Figure 15-2 on page 564. All exchanged commands, replies, and data are text lines delimited by a <CRLF>. All replies have a numeric code at the beginning of the line. The steps of this flow are:

1. The sender SMTP establishes a TCP connection with the destination SMTP and then waits for the server to send a 220 Service ready message or a 421 Service not available message when the destination is temporarily unable to proceed.
2. HELO (HELO is an abbreviation for hello) is sent, to which the receiver will identify itself by sending back its domain name. The sender-SMTP can use this to verify that it contacted the right destination SMTP.

   The sender SMTP can substitute an EHLO command in place of the HELO command. A receiver SMTP that does not support service extensions will respond with a 500 Syntax Error, command unrecognized message. The sender SMTP then retries with HELO, or if it cannot transmit the message without one or more service extensions, it sends a QUIT message.

   If a receiver-SMTP supports service extensions, it responds with a multiline 250 OK message, which includes a list of service extensions that it supports.
3. The sender now initiates the start of a mail transaction by sending a MAIL command to the receiver. This command contains the reverse-path that can be used to report errors. Note that a path can be more than just the user mailbox@host domain name pair. In addition, it can contain a list of routing hosts. Examples of this are when we pass a mail bridge, or when explicit routing information is provided in the destination address. If accepted, the receiver replies with a 250 OK.

4. The second step of the actual mail exchange consists of providing the server SMTP with the destinations for the message. There can be more than one recipient. This is done by sending one or more RCPTTO:<forward-path> commands. Each of them will receive a reply 250 OK if the destination is known to the server, or a 550 No such user here if it is not.
5. When all RCPT commands are sent, the sender issues a DATA command to notify the receiver that the message contents will follow. The server replies with 354 Start mail input, end with <CRLF>.<CRLF>. Note the ending sequence that the sender should use to terminate the message data.
6. The client now sends the data line by line, ending with the 5-character sequence <CRLF>.<CRLF> line, upon which the receiver will acknowledge with a 250 OK, or an appropriate error message if anything went wrong.
7. At this juncture, the client now has several possible actions:
   - If the client has no more messages to send, it can end the connection with a QUIT command, which will be answered with a 221 Service closing transmission channel reply.
   - If the sender has no more messages to send, but is ready to receive messages (if any) from the other side, it can issue the TURN command. The two SMTPs now switch their role of sender/receiver, and the sender (previously the receiver) can now send messages by starting with step 3.
   - If the sender has another message to send, it returns to step 3 and sends a new MAIL command.

Archive

Figure 15-2 illustrates the normal transmission of a single message from a client to a server. Additionally, we provide a textual scenario in Figure 15-3 on page 565.

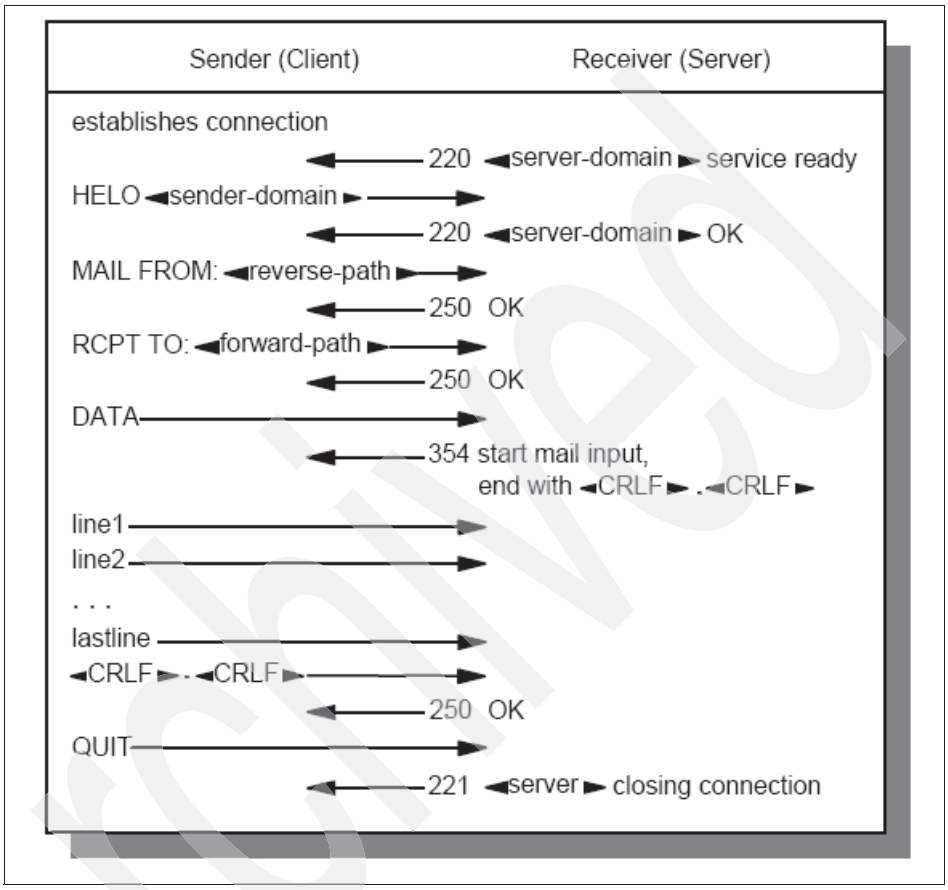


Figure 15-2 SMTP: Normal SMTP data flow - One mail message to one destination mailbox

In the previous description, we mentioned only the most important commands. All of these are commands that must be recognized in each SMTP implementation. Other commands exist, but are optional. RFC standards do not require that every SMTP entity implement them. However, they provide very interesting functions such as relaying, forwarding, mailing lists, and so on. Some of these extensions can be referenced in Table 15-1 on page 557. For a full list of command verbs, refer to RFC 2821.

An additional example

In the following scenario, user abc at host vm1.stockholm.ibm.com sends a note to users xyz, opq, and rst at host delta.aus.edu (Figure 15-3). The lines preceded by *R:* are lines sent by the receiver; the *S:* lines are sent by the sender.

```
R: 220 delta.aus.edu Simple Mail Transfer Service Ready
S: HELO stockholm.ibm.com
R: 250 delta.aus.edu
S: MAIL FROM:<abc@stockholm.ibm.com>
R: 250 OK
S: RCPT TO:<xyz@delta.aus.edu>
R: 250 OK
S: RCPT TO:<opq@delta.aus.edu>
R: 550 No such user here
S: RCPT TO:<rst@delta.aus.edu>
R: 250 OK
S: DATA
R: 354 Start mail input, end with <CRLF>.<CRLF>
S: Date: 23 Jan 89  18:05:23
S: From: Alex B. Carver <abc@stockholm.ibm.com>
S: Subject: Important meeting
S: To:  <xyz@delta.aus.edu>
S: To:  <opq@delta.aus.edu>
S: cc:  <rst@delta.aus.edu>
S:
S: Blah blah blah
S: etc.....
S: .
R: 250 OK
S: QUIT
R: 221 delta.aus.edu Service closing transmission channel
```

Figure 15-3 SMTP an example scenario

Note that the message header is part of the data being transmitted.

15.1.2 SMTP and the Domain Name System

If the network is using the domain concept, an SMTP entity cannot simply deliver mail sent to TEST.MYCORP.COM by opening a TCP connection to

Archived

TEST.MY.CORP. Instead, it must first query the name server to determine to which host (domain name) it needs to deliver the message.

For message delivery, the name server stores resource records (RRs), known as Mail Exchange (MX) RRs. They map a domain name to two values:

- A preference value. Because multiple MX resource records can exist for the same domain name, a preference (priority) is assigned to them. The lowest preference value corresponds to the most preferred record. This is useful whenever the most preferred host is unreachable; the sending SMTP then tries to contact the next preferred host.
- A host name.

It is also possible that the name server responds with an empty list of MX RRs. This means that the domain name is in the name server's authority, but has no MX assigned to it. In this case, the sending SMTP might try to establish the connection with the host name itself.

An important recommendation is given in RFC 2821: After obtaining the MX records, the sending SMTP should query the name server for well-known services (WKS) records for this host and check that the referenced host has SMTP as a WKS entry.

Note: Even though this is only a recommendation, it is widely used in most SMTP implementations.

Here is an example containing MX resource records:

```
mydiv.mycorp.com.       IN    MX 0  smtp1.mydiv.redbookscorp.com.
                        IN    MX 2  smtp2.mydiv.redbookscorp.com.
                        IN    MX 4  smtp3.mydiv.redbookscorp.com.
                        IN    WKS   10.12.34.56 TCP (SMTP)
```

In this example, mail for mydiv.mycorp.com should be delivered to smtp1.mydiv.mycorp.com, because it has a preference of 0. If this host is unreachable, the mail should instead be delivered to the next preferred server, smtp2.mydiv.mycorp.com, which has a preference of 2. If this host is not available, the mail is delivered to the least preferred server, smtp3.mydiv.mycorp.com.

A problem arises when using SMTP and MX records in mixed IPv4 and IPv6 environments. Specifically, a user sending an e-mail does not necessarily know if all of the servers and gateways the message will traverse are all IPv4 or IPv6

capable. As IPv6 becomes more pervasive within the Internet community, the likelihood increases that:

- An IPv4 client will be directed to an IPv6-only server (or vice versa).
- An IPv4 client's message will traverse an IPv6-only network segment (or vice versa).

This impending problem is discussed in RFC 3974, which discusses best practices to circumvent the issue. The primary method for this is ensuring that all IPv6 capable entities also be IPv4 capable.

Addressing mailboxes on server systems

When a user employs a server system for all mail functions, the mailbox address seen by other SMTP users refers exclusively to the mail server system, not to the client system. For example, if two systems are named mydiv.redbookscorp.com and smtp1.mydiv.redbookscorp.com, and the first one is used as a client and the second as a server, the mailbox address might be:

```
user@smtp1.mydiv.redbookscorp.com
```

This mailbox address appears in the *From:* header field of all outgoing mail as well as in the SMTP commands to remote servers issued by the smtp1 server system.

When the user uses a POP server (refer to 15.4, "Post Office Protocol (POP)" on page 589), however, the mailbox address on outbound mail items contains the workstation's host name (for example, user@mydiv.redbookscorp.com). In this case, the sender should include a *Reply-To:* field in the mail header to indicate that replies should *not* be sent to the originating mailbox. For example, the mail header might look like this:

```
Date: Fri, 10 Feb 95 15:38:23
From: user@mydiv.redbookscorp.com
To: "Your Email" <youremail@yourdiv.redbookscorp.com>
Reply-To: user@smtp1.mydiv.redbookscorp.com
Subject: Another sample email
```

Note that the From: address reflects the address of the user's client system, while the Reply-To: field reflects the address of the smtp1 server system. This is because messages sent to the client system will fail because no SMTP server is active there. Instead, messages sent back to the user will be sent to the Reply-To address, and the user's client can then connect to the smtp1 server to retrieve its mail. For this reason, the receiving mail agent is expected to send replies to the Reply-To: address and not the From: address.

Using the Domain Name System to direct mail

An alternative approach to using the Reply-To: header field is to use the Domain Name System to direct mail to the correct mailbox. The administrator for the domain name server (with authority for the domain containing the user's workstation and the name server) can add MX resource records to the Domain Name System to direct mail appropriately, as described in 15.1.2, “SMTP and the Domain Name System” on page 565. For example, the following MX records indicate to the client SMTPs that if the SMTP server for the mydiv.redbookscorp.com domain (smtp1.mydiv.redbookscorp.com) is not available, there is a mail server on smtp2.yourdiv.redbookscorp.com (10.12.34.56) that should be used instead:

```
smtp2.yourdiv.redbookscorp.com. IN     WKS   10.12.34.56 TCP (SMTP)
mydiv.redbookscorp.com.         IN     MX 0  smtp1.mydiv.redbookscorp.com.
                              IN     MX 1  smtp2.yourdiv.redbookscorp.com.
```

15.2 Sendmail

Sendmail is a command-line tool designed for most UNIX-like operating systems. It does not define a method of transferring mail, but rather acts as a client/server that supports multiple mail protocols. One of the oldest mail transfer agents (MTAs) on the Internet, sendmail descends from the original ARPANET *delivermail* application. Currently, it exists in both open source and proprietary software packages.

15.2.1 Sendmail as a mail transfer agent (MTA)

The model on which most mailing systems are built consist of three types of components:

- Mail user agent (MUA)
  The MUA is the interface through which a user can read and send email.
- Mail transfer agent (MTA)
  The MTA acts like a mail router, accepting messages from both MTAs and MUAs. Based on the message headers, the MTA decides what delivery method should be used and passes the message to the correct mail delivery agent.
- Mail delivery agent (MDA)
  The MDA accepts messages from MTAs, and delivers the message to the appropriate location. You can think of this as a transport layer for the mail model.

Figure 15-4 provides a diagram of how these components interact.

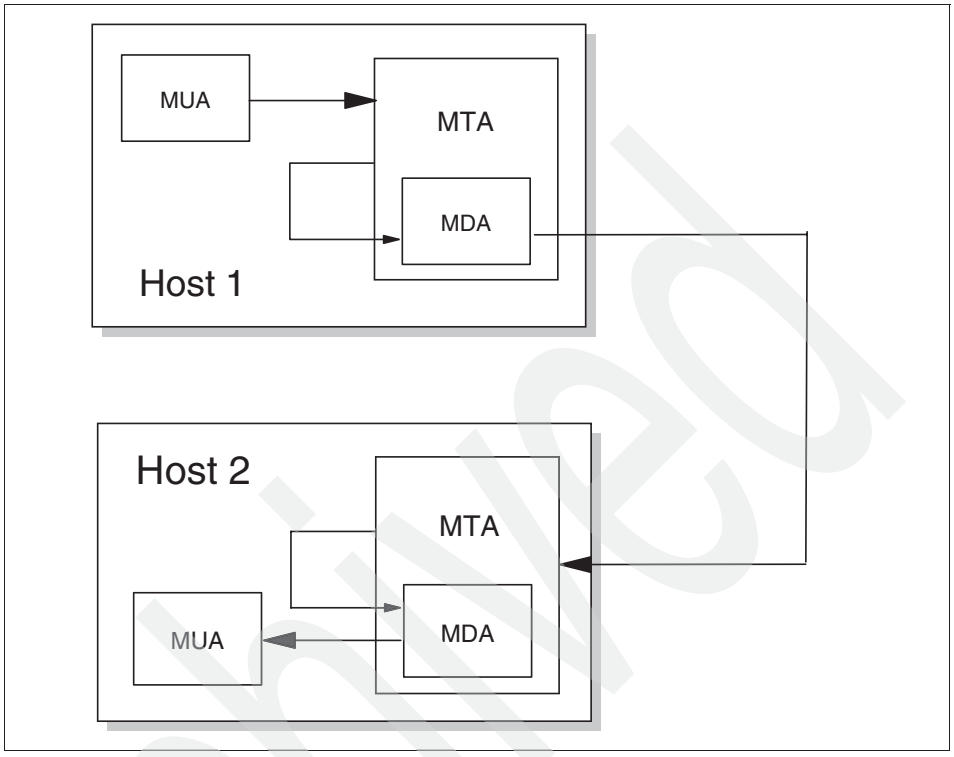


Figure 15-4 The typical mail model

Within this model, sendmail plays the role of the MTA. While most current implementations of sendmail now employ SMTP (see 15.1, “Simple Mail Transfer Protocol” on page 556), the application has the flexibility to use any mail protocol—standard or proprietary.

15.2.2 How sendmail works

Sendmail uses a queuing system to manage inbound and outbound mail. Running as a daemon, sendmail listens on well-known port 25 for inbound messages. Upon receiving a message, sendmail spawns a process to queue the message up in a queue file. Though the location of this file is implementation specific, it usually exists as /var/spool/mqueue.

If the message can be delivered immediately, the spawned process does so, de-queues the message, and then terminates. If the message cannot be delivered, the process leaves the message on the queue, but still terminates. At regular intervals, the sendmail parent process spawns a new process to check

the queue file, attempts to send any messages still on the queue, and then terminates. Messages that remain on the queue for too long will ultimately be discarded.

This process is illustrated in Figure 15-5.

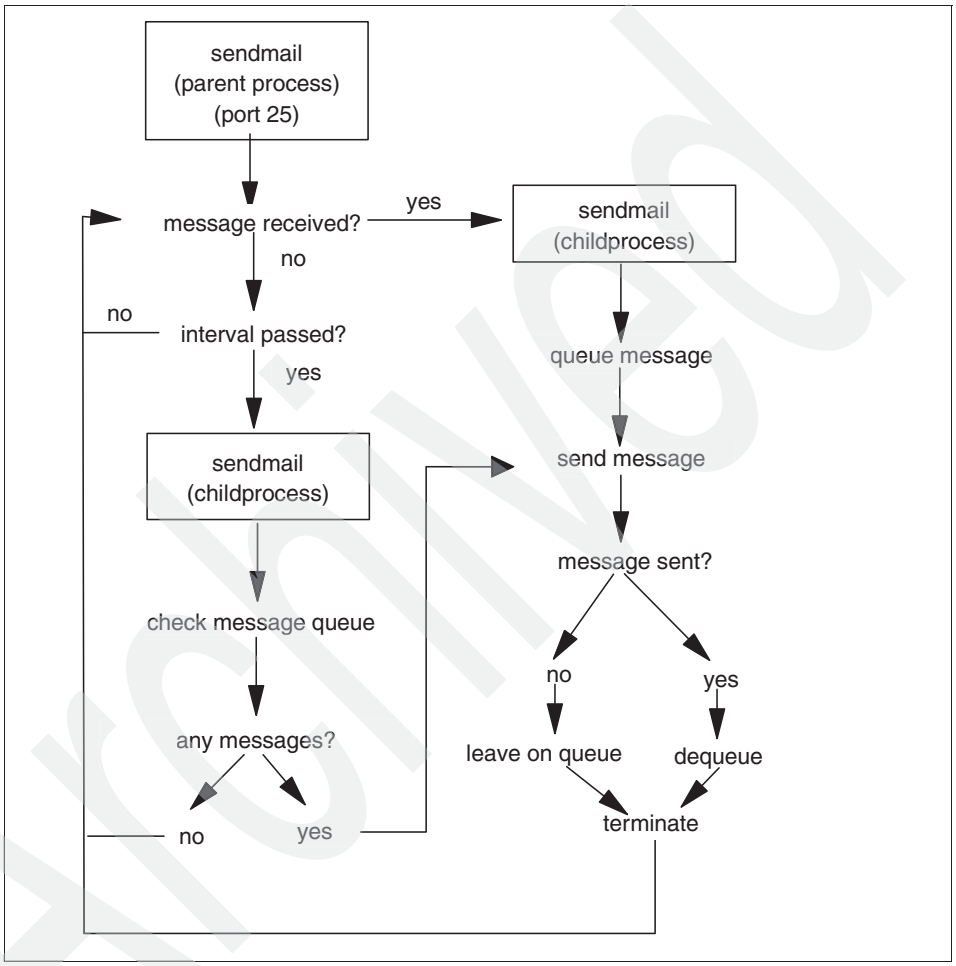


Figure 15-5 Message processing in sendmail

The open source version of sendmail is maintained by the Sendmail Consortium, which releases regular updates of the sendmail source code[1]. However, the consortium is sponsored by Sendmail, Inc., which maintains a proprietary sendmail software package.

[1] At the time of writing, the most recent version is Sendmail 8.13.7, released April 14, 2006.

For additional information, refer to:

http://www.sendmail.org

15.3 Multipurpose Internet Mail Extensions (MIME)

Electronic mail (as described in 15.1, “Simple Mail Transfer Protocol” on page 556) is probably the most widely used TCP/IP application. However, SMTP is limited to 7-bit ASCII text, with a maximum line length of 1000 characters. This results in a number of limitations, including:

- SMTP cannot transmit executable files or other binary objects. There are ad hoc methods of encapsulating binary items in SMTP mail items, such as:
  - Encoding the file as pure hexadecimal
  - The UNIX **uuencode** and **uudecode** utilities, used to encode binary data in the UNIX-to-UNIX Copy (UUCP) mailing system

  However, none of these can be described as a *de facto* standard (though **uuencode**/**uudecode** is perhaps the most pervasive, due to the pioneering role of UNIX systems in the Internet).
- SMTP cannot transmit text data that includes national language characters, because these are represented by code points with a value of 128 (decimal) or higher in all character sets based on ASCII.
- SMTP servers might reject mail messages over a certain size. Any given server can have permanent or transient limits, or both, on the maximum amount of mail data it can accept from a client at any given time.
- SMTP gateways that translate from ASCII to EBCDIC and vice versa do not use a consistent set of code page mappings, resulting in translation problems.
- Some SMTP implementations or other mail transport agents (MTAs) in the Internet do not adhere completely to the SMTP standards defined in RFC 2821. Common problems include:
  - Removal of trailing white-space characters (tabs and spaces).
  - Addition of white-space characters to make all lines in a message the same length.
  - Wrapping of lines longer than 76 characters.
  - Changing of new line sequences between different conventions. (For example, <CRLF> characters might be converted to <CR> or <LF> sequences.)
  - Conversion of tab characters to multiple spaces.

MIME is a draft standard that includes mechanisms to resolve these problems in a manner that is highly compatible with existing RFC 2822 standards. Because mail messages are frequently forwarded through mail gateways, it is not possible for an SMTP client to distinguish between a server that manages the destination mailbox and one that acts as a gateway to another network. Because mail that passes through a gateway might be tunnelled through further gateways, some or all of which can be using a different set of messaging protocols, it is not possible in general for a sending SMTP to determine the lowest common denominator capability common to all stages of the route to the destination mailbox. For this reason, MIME assumes the worst: 7-bit ASCII transport, which might not strictly conform to or be compatible with RFC 2821. It does not define any extensions to RFC 2821, but limits itself to extensions within the framework of RFC 2822. Therefore, a MIME message is one which can be routed through any number of networks that are loosely compliant with RFC 2821 or are capable of transmitting RFC 2821 messages.

MIME can be described in five parts:

- Protocols for including objects other than US ASCII text mail messages within the bodies of messages conforming to RFC 2822. These are described in RFC 2045.
- General structure of the MIME media typing system, which defines an initial set of media types. This is described in RFC 2046.
- A protocol for encoding non-U.S. ASCII text in the header fields of mail messages conforming to RFC 2822. This is described in RFC 2047.
- Various IANA registration procedures for MIME-related facilities. This is described in RFC 2048.
- MIME conformance criteria. This is described in RFC 2049.

Although RFC 2045 provides a mechanism suitable for describing non-textual data from X.400 messages in a form that is compatible with RFC 2822, it does not say how X.400 message parts are to be mapped to MIME message parts. The conversion between X.400 and MIME is defined in RFCs 2156 and 2157, which update the protocols for the conversion between RFC 822 (obsoleted by RFC 2822) and X.400.

Archive

The MIME standard was designed with the following general order of priorities:

1. Compatibility with existing standards, such as RFC 2822.

   There are two areas where compatibility with previous standards is not complete:

   – RFC 1049 (which is part of STD 11) described a *Content-Type:* field used to indicate the type of (ASCII text) data in a message body. PostScript or SGML allows a user mail agent to process it accordingly. MIME retains this field, but changes the values that are defined for it. Because the correct response for a mail agent on encountering an unknown value in this field is basically to ignore it, this does not raise any major compatibility concerns.

   – RFC 934 discusses encapsulation of messages in the context of message forwarding and defines encapsulation boundaries, or lines indicating the beginning and end of an encapsulated message. MIME retains broad compatibility with RFC 934, but does not include the quoting mechanism used by RFC 934 for lines in encapsulated messages that could otherwise be misinterpreted as boundaries.[2]

   The most important compatibility issue is that the standard form of a MIME message is readable with an RFC 2821-compliant mail reader. In particular, the default encoding for MIME message bodies is to implement no encoding at all, just like RFC 2822.

2. The next priority was to employ robustness across existing practice. As noted earlier, there are many widely deployed MTAs in the Internet that do not comply with STD 10/RFC 2821. The encoding mechanisms specified in RFC 2045 are designed to always circumvent the most common of these (folding of lines as short as 76 characters and corruption of trailing white space characters) by only transmitting short lines with no trailing white space characters and allowing encoding of any data in a mail safe fashion.

[2] The reason for this departure is that MIME allows for deeply nested encapsulation, but encodes text in such a way as to reversibly spill text lines at or before column 76 to avoid the lines being spilled irreversibly by non-conforming SMTP agents. The RFC 934 quoting mechanism can result in lines being lengthened with each level of encapsulation, possibly past column 76.

Note: MIME does not require mail items to be encoded; the decision is left to the user or the mail program, or both. For binary data transmitted across (7-bit) SMTP, encoding is invariably required, but for data consisting mostly of text, this might not be the case.

The preferred encoding mechanism for mostly text data is that, at a minimum, it is mail-safe with any compliant SMTP agent on an ASCII system, and at maximum, is mail-safe with all known gateways and MTAs. The reason why MIME does not require maximum encoding is that the encoding hampers readability when the mail is transmitted to non-MIME-compliant systems.

3. The priority considered when designing MIME was ease of extension. RFC 2045 categorizes elements of mail bodies into seven *content-types*, which have *subtypes*. The content-type/subtype pairs in turn have parameters that further describe the object concerned. The RFC defines a mechanism for registering new values for these and other MIME fields with the Internet Assigned Numbers Authority (IANA). This process is itself updated by RFC 2048.

   For the current list of all MIME values, consult STD 2 – Assigned Internet Numbers. The remainder of this section describes only the values and types given in RFC 2045.

One consequence of this approach is that, to quote RFC 2045, “some of the mechanisms [used in MIME] may seem somewhat strange or even baroque at first. In particular, compatibility was always favored over elegance.”

Because RFC 2822 defines the syntax of message headers (and deliberately allows for additions to the set of headers it describes) but not the composition of message bodies, the MIME standard is largely compatible with RFC 2822. This is particularly true for the portion of RFC 2045 which defines the structure of message bodies, as well as a set of header fields used to describe that structure.

MIME can be seen as a high-level protocol; since it works entirely within the boundaries of STD 10 and STD 11, it does not involve the transport layer (or lower layers) of the protocol stack at all.

15.3.1 How MIME works

A MIME-compliant message must contain a header field with the following verbatim text:

```
MIME-Version: 1.0
```

As with RFC 2822 headers, the case of MIME header field names are never significant, but the case of field values can be, depending on the field name and the context. For the MIME fields described later, the values are case-insensitive unless stated otherwise.

The general syntax for MIME header fields is the same as that for RFC 2822, having the format:

```
Keyword: Value
```

Therefore, the following field is valid (parenthetical phrases are treated as comments and ignored):

```
MIME-Version: 1.0 (this is a comment)
```

Table 15-3 lists and defines the five MIME headers.

Table 15-3 MIME header fields

Keyword	Value
MIME-Version	As noted earlier, this must have the value 1.0.
Content-Type	This describes how the object within the body is to be interpreted. The default value is text/plain. The default character set parameter for this is the charset=us-ascii, which indicates unformatted 7-bit ASCII text data.
Content-Transfer-Encoding	This describes how the object within the body was encoded in order that it be included in the message using a mail-safe form.
Content-Description	A plain-text description of the object within the body, which is useful when the object is not human-readable (such as audio or image data).
Content-ID	A world-unique value identifying the content of this part of this message.

The second and third of these fields are described in more detail in the following sections.

The Content-Type field

The body of the message is described with a Content-Type field of the form:

```
Content-Type: type/subtype ;parameter=value ;parameter=value
```

The allowable parameters are dependent on the type and subtype. Some type/subtype pairs have no parameters, some have optional ones, some have

Archived

mandatory ones, and some have both. The subtype parameter *cannot* be omitted, but the whole field can, in which case the default value is text/plain.

There are seven standard content-types:

- Text

  A single subtype, *plain*, is defined for the text type, specifying unformatted text. A parameter can optionally be included with this type/subtype pair in order to specify the character set of the text. The following values are permitted for this parameter:

  - us-ascii: The text consists of ASCII characters in the range 0 to 127 (decimal). This is the default (for compatibility with RFC 2822).
  - iso-8859-x: Where x is in the range 1 to 9 for the different parts of the ISO-8859 standard. The text consists of ISO characters in the range 0 to 255 (decimal). All of the ISO-8859 character sets are ASCII-based with national language characters and other special characters in the range 128 to 255. Note that if the text contains no characters with values above 127, the character set is specified as us-ascii, because it can be adequately represented in that character set.
  - Format: The data consists of fixed or flowed text. When specified, an additional parameter of fixed or flowed can be specified to indicate the exact nature of the text. If no parameter is included, fixed is assumed.
  - DelSp: Indicates if trailing whitepace in fixed or flowed text should be preserved or deleted. Its values are yes or no, and if nothing is specified no is assumed.

  Noted that us-ascii and iso-8859-x were initially defined for MIME in RFC 2046, while Format and DelSp were added by RFC 3676. Further subtypes can be added to describe other readable text formats (such as word processor formats) that contain formatting information for an application to enhance the appearance of the text.

- Multipart

  The message body can contain multiple objects of independent data types. In each case, the body is divided into parts by lines called *encapsulation boundaries*. The contents of the boundary are defined with a parameter in the content-type field, for example:

  ```
  Content-Type: multipart/mixed; boundary="1995021309105517"
  ```

  The boundary must not appear in any of the parts of the message. It is case-sensitive and consists of 1-70 characters from a set of 75 that are known to be very robust through mail gateways, and it cannot end in a space. (The example uses a 16-digit decimal time stamp.) Each encapsulation boundary consists of the boundary value prefixed by a <CRLF> sequence

and two hyphens (for compatibility with RFC 934). The final boundary that marks the end of the last part also has a suffix of two hyphens. Within each part, there is a MIME header, which, like ordinary mail headers, is terminated by the sequence <CRLF><CRLF> but can be blank. The header fields define the content of the encapsulated message.

Four subtypes are defined:

- Mixed: The different parts are independent but are transmitted together. They must be presented to the recipient in the order that they appear in the mail message.
- Parallel: This differs from the mixed subtype only in that no order is ascribed to the parts. Therefore, the receiving mail program can, for example, display all of them in parallel.
- Alternative: The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system displays the best version to the user.
- Digest: This is a variant on multipart/mixed where the default type/subtype is message/rfc822 instead of text/plain. It is used for the common case where multiple RFC 2822 or MIME messages are transmitted together. In this case, the body is an encapsulated message, or part of one. Three possible subtypes are defined:
  - rfc822: The body itself is an encapsulated message with the syntax of an RFC 2822 message. It is required that at least one of From:, Subject:, or Date: be present.

    Note: Note that, though RFC 822 was obsoleted by RFC 2822, the message subtype used is still 822.

  - partial: This type is used to allow fragmentation of large mail items in a similar way to IP fragmentation. Because SMTP agents can impose upper limits on maximum mail sizes, this might be necessary to send large items. The intent of the message/partial mail items is that the fragmentation is transparent to the recipient. The receiving user agent should reassemble the fragments, creating a new message with semantics identical to the original. There are three parameters for the Content-Type: field:

    id= A unique identifier common to all parts of the message.

    number= The sequence number of this part, with the first part being numbered 1.

total= The total number of parts. This is optional on all but the last part. The last part is identified by the fact that it has the same value for the number and total parameters.

The original message is always adheres to RFC 2822's rules. The first part is syntactically equivalent to a message/RFC 822 message (that is, the body itself contains message headers), and the subsequent parts are syntactically equivalent to text/plain messages. When rebuilding the message, the RFC 2822 header fields are taken from the top-level message, not from the enclosed message. The exceptions to this are those fields that cannot be copied from the inner message to the outer when fragmentation is performed (for example, the Content-Type: field).

- external-body: This type contains a pointer to an object that exists elsewhere. It has the syntax of the message/RFC 822 type. The top-level message header defines how the external object is to be accessed, using the access-type: parameter of the Content-Type: field and a set of additional parameters that are specific to the access type. The intent is for the mail reader to be able to synchronously access the external object using the specified access type. The following access types are defined:

  ftp File Transfer Protocol. The recipient is expected to supply the necessary user ID and password. For security reasons, these are never transmitted with the message.

  tftp Trivial File Transfer Protocol.

  anon-ftp Anonymous FTP.

  local-file The data is contained in a file accessible directly through the recipient's local file system.

  mail-server The data is accessible through a mail server. Unlike the others, this access is necessarily asynchronous.

  When the external object has been received, the desired message is obtained by appending the object to the message header encapsulated within the body of the message/external-body message. This encapsulated message header defines how the resulting message is to be interpreted. (It is required to have a Content-ID: and will normally have a Content-Type: field.) The encapsulated message body is not used (the real message body is elsewhere, after all) and it is therefore termed the *phantom body*. There is one exception to this: If the access-type is mail-server, the phantom body contains the mail server commands necessary to extract the real message body. This is

because mail server syntaxes vary widely, so it is much simpler to use the otherwise redundant phantom body than to codify a syntax for encoding arbitrary mail server commands as parameters on the Content-Type: field.

An example of a complex multipart message is shown in Figure 15-6, and continued in Figure 15-7 on page 580.

```
MIME-Version: 1.0
From: My Email <myemail@mydiv.redbookscorp.com>
To: Your Email <youremali@mydiv.redbookscorp.com>
Subject: Multipart message
Content-type: multipart/mixed; boundary="1995021309105517"

This section is called the preamble. It is after the header but before the first
boundary. Mail readers which understand multipart messages must ignore this.
--1995021309105517

The first part. There is no header, so this is text/plain with
charset=us-ascii by default. The immediately preceding <CRLF> is part of the
<CRLF><CRLF> sequence that ends the null header. The one at the end is part of the
next boundary, so this part consists of five lines of text with four <CRLF>s.
--1995021309105517
Content-type: text/plain; charset=us-ascii
Comments: this header explicitly states the defaults

One line of text this time, but it ends in a line break.

--1995021309105517
Content-Type: multipart/alternative; boundary=_
Comments: An encapsulated multipart message!

Again, this preamble is ignored. The multipart body contains a still image and a
video image encoded in Base64. See 11.2.3.5, “Base64 encoding” on page 413. One
feature is that the character "_" which is allowed in multipart boundaries never
occurs in Base64 encoding so we can use a very simple boundary!
--_
Content-type: text/plain

This message contains images which cannot be displayed at your terminal.
This is a shame because they're very nice.
--_
Content-type: image/jpeg
Content-transfer-encoding: base64
Comments: This photograph is to be shown if the user's system cannot display MPEG
```

Figure 15-6 MIME: A complex multi-part example

```
videos. Only part of the data is shown in this book
because the reader is unlikely to be wearing MIME-compliant spectacles.

Qk1OAAAAAAAAE4EAABAAAAAQAEAAPAAAAABAAgAAAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAAAAA
AAAAAAAAAAAAAAAAAAAAAAB4VjQSAAAAAAAAgAAAkgAAAJKAAKoAAACqAIAAqpIAAMHBwQDJyckA
/9uqAKpJAAD/SQAAAGOAAFVtAACqbQAA/2OAAAAkAABVkgAAqiQAAP+SAAAAtgAAVbYAAKq2AAD/
<base64 data continues for another 1365 lines>
--_
Content-type: video/mpeg
Content-transfer-encoding: base64

AAABswoAeBn//+CEAAABsgAAAOgAAAG4AAAAAAAAAQAAT/////wAAAGy//8AAAEBQ/ZlIwwBGWCX
+pqMiJQDjAKywS/1NRrtXcTCLgzVQymqqHAfOsL1sMgMq4SWLCwOTYRdgyAyrhNYsLhhF3DLjAGg
BdwDXBv3yMV8/4tzrp3zsAWIGAJg1IBKTeFFI2IsgutIdfuSaAGCTsBVnWdz8afdMMAMgKgMEkPE
<base64 data continues for another 1839 lines>
--_--
That was the end of the nested multipart message. This is the epilogue.
Like the preamble it is ignored.
--1995021309105517--
And that was the end of the main multipart message. That's all folks!
```

Figure 15-7 MIME: A complex multi-part example, continued

- Image

  The body contains image data requiring a graphical display or some other device, such as a printer, to display it. Two subtypes are defined initially:

  - jpeg: The image is in JPEG format, JFIF encoding.
  - gif: GIF format.

- Video

  The body contains moving image data (possibly with synchronized audio) requiring an intelligent terminal or multimedia workstation to display it. A single subtype is defined initially:

  - mpeg: MPEG format.

- Audio

  The body contains audio data requiring a speaker and sound card (or similar hardware) to display it. A single subtype is defined initially:

  - basic: A lowest common denominator format in the absence of any de facto standards for audio encoding. Specifically, it is single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.

Archived

- Application

  This type is intended for types that do not fit into other categories, and particularly for data to be processed by an application program before being presented to the user, such as spreadsheet data. It is also intended for application programs that are intended to be processed as part of the mail reading process (for example, see the PostScript type). This type of usage poses serious security risks unless an implementation ensures that executable mail messages are run in a safe or *padded cell* environment.

  Two subtypes are defined initially:

  – PostScript: Adobe Systems PostScript (Level 1 or Level 2)

    PostScript security issues: Although PostScript is often thought of as a format for printer data, it is a programming language and the use of a PostScript interpreter to process application/PostScript types poses serious security problems. Any mail reader that automatically interprets PostScript programs is equivalent, in principle, to one that automatically runs executable programs it receives. RFC 2046 outlines the issues involved.

  – octet-stream

    This subtype indicates general binary data consisting of 8-bit bytes. It is also the subtype that a mail reader assumes on encountering an unknown type or subtype. Any parameters are permitted, and RFC mentions two: a *type=* parameter to inform the recipient of the general type of the data, and *padding=* to indicate a bit stream encoded in a byte stream. (The padding value is the number of trailing zero bits added to pad the stream to a byte boundary.)

    Implementations are recommended to offer the user the option of using the data as input to a user program or storing it in a file. An optional *Content-Disposition:* field, described in RFC 2183, allows the specification of the preferred name of such a file.

    Security issues: The RFCs strongly recommend against an implementation automatically executing an application/octet-stream part or using it as input to a program specified in the mail header. To do so exposes the receiving system to serious security risks and might impact the integrity of any networks to which the system is connected.

Obviously, there are many types of data that do not fit into any of the previous subtypes. Cooperating mail programs can, in keeping with the rules of RFC 2822, use types or subtypes beginning with X- as private values. No other values are permitted unless they have first been registered with the Internet Assigned Numbers Authority (IANA). See RFC 2048 for more details. The intention is that few, if any, additional types will be needed, but that many subtypes will be added to the set.

One such addition defined in RFC 3798. This extends the message type with a disposition-notification subtype. This subtype allows a mail user agent or an electronic mail gateway to return notifications to a sender indicating the disposition of a sent message, emulating a functionality often found in X.400 and proprietary LAN-based networks.

15.3.2 The Content-Transfer-Encoding field

As already noted, SMTP agents and mail gateways can severely constrain the contents of mail messages that can be transmitted safely. The MIME types described earlier list a rich set of different types of objects that can be included in mail messages, and the majority of these do not fall within these constraints. Therefore, it is necessary to encode data of these types in a fashion that can be transmitted and to decode them on receipt. RFC 2045 defines two forms of encoding that are mail safe. The reason for two forms rather than one is that it is not possible, given the small set of characters known to be mail safe, to devise a form that can both encode text data with minimal impact to the readability of the text and yet can encode binary data that consists of characters distributed randomly across all 256 byte values compactly enough to be practical.

These two encodings are used only for bodies and not for headers. We describe header encoding in 15.3.3, “Using non-ASCII characters in message headers” on page 587. The *Content-Transfer-Encoding:* field defines the encoding used. Although cumbersome, this field name emphasizes that the encoding is a feature of the transport process and not an intrinsic property of the object being mailed. Although there are only two encodings defined, this field can take on *five* values. (As usual, the values are case-insensitive.) Three of the values specify that no encoding has been done; where they differ is that they imply different reasons for why this is the case. This is a subtle but important point. MIME is not restricted to SMTP as a transport agent, despite the prevalence of (broadly) SMTP-compliant mail systems on the Internet. It therefore allows a mail agent to transmit data that is not mail-safe by the standards of SMTP (that is, STD 10/RFC 2821). If such a mail item reaches a gateway to a more restrictive system, the encoding mechanism specified allows the gateway to decide on an item-by-item basis whether the body must be encoded to be transmitted safely.

The five encodings are:

- 7-bit (the default if the Content-Transfer-Encoding: header is omitted)
- 8-bit
- Binary
- Quoted-Printable
- Base64

We describe these in the sections that follow.

7-bit encoding

Seven-bit encoding means that no encoding has been done, and the body consists of lines of ASCII text with a length of no more than 1000 characters. It is therefore known to be mail-safe with any mail system that *strictly* conforms with STD 10/RFC 2821. This is the default, because these are the restrictions that apply to pre-MIME STD 11/RFC 2822 messages.

Note: Seven-bit encoding does *not* guarantee that the contents are truly mail safe for two reasons. First, gateways to EBCDIC networks have a smaller set of mail-safe characters, and secondly because of the many non-conforming SMTP implementations. The Quoted-Printable encoding is designed to overcome these difficulties for text data.

8-bit encoding

Eight-bit encoding implies that lines are short enough for SMTP transport, but that there might be non-ASCII characters (that is, octets with the high-order bit set). Where SMTP agents support the SMTP service extension for 8-bit-MIMEtransport, described in RFC 1652, 8-bit encoding is possible. Otherwise, SMTP implementations must set the high-order bit to zero, so 8-bit encoding is not valid.

Binary encoding

Binary encoding indicates that non-ASCII characters might be present and that the lines might be too long for SMTP transport. (That is, there might be sequences of 999 or more characters without a <CRLF> sequence.) There are currently no standards for the transport of un-encoded binary data by mail based on the TCP/IP protocol stack, so the only case where it is valid to use binary encoding in a MIME message sent on the Internet or other TCP/IP-based network is in the header of an external-body part (see the message/external-body type earlier). Binary encoding would be valid if MIME were used in conjunction with other mail transport mechanisms, or with a hypothetical SMTP service extension that did support long lines.

Quoted-Printable encoding

This is the first of the two real encodings and it is intended to leave text files largely readable in their encoded form. Quoted-Printable encoding:

- Represents non-mail safe characters by the hexadecimal representation of their ASCII characters.
- Introduces reversible (soft) line breaks to keep all lines in the message to a length of 76 characters or less.

Quoted-Printable encoding uses the equal sign as a quote character to indicate both of these cases. It has five rules, which are summarized as follows:

- Any character except one that is part of a new line sequence (that is, a X'0D0A' sequence on a text file) can be represented by =XX, where XX are two uppercase hexadecimal digits. If none of the other rules apply, the character must be represented as XX.
- Any character in the range X'21' to X'7E', except for X'3D' (=), can be represented as the ASCII character.
- ASCII tab (X'09') and space (X'20') can be represented as the ASCII character, except when it is the last character on the line.
- A line break must be represented by a <CRLF> sequence (X'0D0A'). When encoding binary data, X'0D0A' is not a line break must should be coded, according to rule 1, as =0D=0A.
- Encoded lines cannot be longer than 76 characters (excluding the <CRLF>). If a line is longer than this, a soft line break must be inserted at or before column 75. A soft line break is the sequence =<CRLF> (X'3D0D0A').

This scheme is a compromise between readability, efficiency, and robustness. Because rules 1 and 2 use the phrase “may be encoded,” implementations have a fair degree of latitude on how many characters are quoted. If as few characters are quoted as possible within the scope of the rules, then the encoding will work with well-behaved ASCII SMTP agents. Adding the following set of ASCII characters to the list of those to be quoted is adequate for well-behaved EBCDIC gateways:

```
! " # $ @ [ \ ] ^ ` { | } ~
```

For total robustness, it is better to quote *every* character except for the 73-character set known to be invariant across all gateways, that is the letters and digits (A-Z, a-z and 0-9) and the following 11 characters:

```
' ( ) + , - . / : = ?
```

Note: This invariant list does not even include the space character. For practical purposes, when encoding text files, only a space should be quoted. Otherwise, at the end of a line, readability is severely impacted.

Base64 encoding

This encoding is intended for data that does not consist mainly of text characters. Quoted-Printable encoding replaces each non-text character with a 3-byte sequence, which is grossly inefficient for binary data. Base64 encoding works by treating the input stream as a bit stream, regrouping the bits into shorter bytes, padding these short bytes to 8 bits, and then translating these bytes to

characters that are known to be mail-safe. As noted in the previous section, there are only 73 safe characters, so the maximum byte length usable is 6 bits, which can be represented by 64 unique characters (thus the name Base64). Because the input and output are both byte streams, the encoding has to be done in groups of 24 bits (that is 3 input bytes and 4 output bytes). The process can be seen as shown in Figure 15-8.

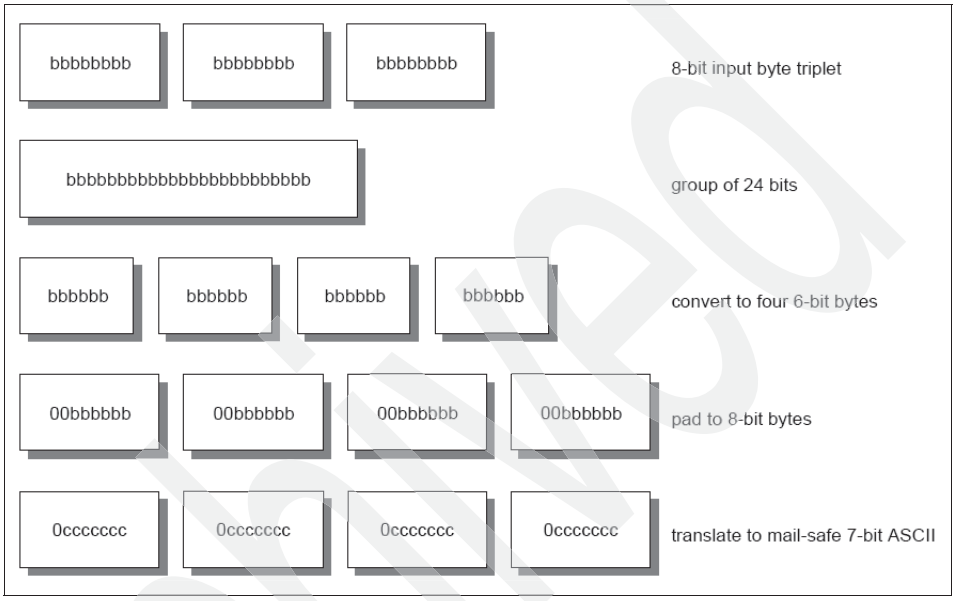


Figure 15-8 MIME: Base64 encoding - How 3 input bytes are converted to 4 output bytes in the Base64 encoding scheme

The translate table used is called the Base64 alphabet, as shown in Table 15-4.

Table 15-4 The Base64 alphabet

Base64 value	ASCII char	Base64 value	ASCII char	Base64 value	ASCII char	Base64 value	ASCII char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1

Base64 value	ASCII char	Base64 value	ASCII char	Base64 value	ASCII char	Base64 value	ASCII char
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

One additional character (the = character) is needed for padding. Because the input is a byte stream that is encoded in 24-bit groups, it will be short by zero, 8, or 16 bits, as will the output. If the output is of the correct length, no padding is needed. If the output is 8 bits short, this corresponds to an output quartet of two complete bytes, a short byte, and a missing byte. The short byte is padded with two low-order zero bits. The missing byte is replaced with an = character. If the output is 16 bits short, this corresponds to an output quartet of one complete byte, a short byte, and two missing bytes. The short byte is padded with six low-order zero bits. The two missing bytes are replaced with an = character. If zero characters (that is, As) were used, the receiving agent would not be able to tell, when decoding the input stream, if the trailing X'00' characters in the last or last two positions of the output stream were data or padding. With pad characters, the number of “=”s (0, 1, or 2) gives the length of the input stream modulo 3 (0, 2, or 1, respectively).

Conversion between encodings

The Base64 encoding can be freely translated to and from the binary encoding without ambiguity, because both treat the data as an octet-stream. This is also true for the conversion from Quoted-Printable to either of the other two (in the case of the Quoted-Printable to binary conversion, the process can be viewed as involving an intermediate binary encoding) by converting the quoted character sequences to their 8-bit form, deleting the soft line breaks, and replacing hard line breaks with <CRLF> sequences. This is not strictly true of the reverse process, because Quoted-Printable is a record-based system. There is a semantic difference between a hard line break and an imbedded =0D=0A

sequence. (For example, when decoding Quoted-Printable on an EBCDIC record-based system such as VM, hard line breaks map to record boundaries, but =0D=0A sequences map to X'0D25' sequences.)

Multiple encodings

MIME does *not* allow nested encodings. Any Content-Type that recursively includes other Content-Type fields (notably the multipart and message types) cannot use a Content-Transfer-Encoding other than 7-bit, 8-bit, or binary. All encodings must be done at the innermost level. The purpose of this restriction is to simplify the operation of user mail agents. If nested encodings are not permitted, the structure of the entire message is always visible to the mail agent without the need to decode the outer layer or layers of the message.

This simplification for user mail agents has a price: complexity for gateways. Because a user agent can specify an encoding of 8-bit or binary, a gateway to a network where these encodings are not safe must encode the message before passing it to the second network. The obvious solution, to simply encode the message body and to change the Content-Transfer-Encoding: field, is not allowed for the multipart or message types, because it violates the restriction described earlier. The gateway must therefore correctly parse the message into its components and re-encode the innermost parts as necessary.

There is one further restriction: Messages of type message/partial must *always* have 7-bit encoding. (Eight-bit and binary are also disallowed.) The reason for this is that if a gateway needs to re-encode a message, it requires the entire message to do so, but the parts of the message might not all be available together. (Parts might be transmitted serially because the gateway is incapable of storing the entire message at once, or they might even be routed independently through different gateways.) Therefore, message/partial body parts must be mail safe across lowest common denominator networks; that is, they must be 7-bit encoded.

15.3.3 Using non-ASCII characters in message headers

All of the previous mechanisms refer exclusively to bodies and not to headers. The contents of message headers must still be coded in US-ASCII. For header fields that include human-readable text, this is not adequate for languages other than English. A mechanism to include national language characters is defined by the second part of MIME (defined in RFC 2047, and extended in RFC 2231). This mechanism differs from the Quoted-Printable encoding, which is used in a message body for the following reasons:

- The format of message headers is strictly codified by RFC 2822, so the encoding used by MIME for header fields must work within a narrower set of constraints than that used for bodies.

- Message relaying programs frequently change message headers. For example: re-ordering header fields, deleting some fields but not others, re-ordering mailboxes within lists, or spilling fields at different positions than the original message.
- Some message handling programs do not correctly handle some of the more arcane features of RFC 2822 (such as the use of the \ character to quote special characters, such as < and >).

The approach used by MIME is to reserve improbable sequences of legal ASCII characters that are not syntactically important in RFC 2822 for use with this protocol. Words in header fields that need national characters are replaced by *encoded words*, which have the form:

```
=?charset?encoding?word?=
```

Where:

charset The value allowed for the charset parameter used with text/plain MIME type, that is “us-ascii”, or “iso-8859-1” through “iso-8859-9”.

encoding B or Q. B is identical to the Base64 encoding used in message bodies. Q is similar to the Quoted-Printable encoding but uses an underscore (_) to represent X' 20' (ASCII space).[3] Q encoding requires the encoding of _ characters and does not allow line breaks. Any printable ASCII character other than _, =, and space can be left un-quoted within an encoded word unless it is syntactically meaningful when the header field is parsed according to RFC 2822.

Charset and encoding are both case-insensitive.

word A string of ASCII text characters other than space, which conforms to the rules of the encoding given.

An encoded word must have no imbedded white space characters (space or tab), can be up to 75 characters long, and cannot be on a line that is greater than 76 characters long (excluding the <CRLF>). These rules ensure that gateways will not fold encoded words in the middle of the word. Encoded words can generally be used in the human-readable parts of header fields. For example, if a mailbox is specified in the following form:

```
Your Email <youremail@yourdiv.redbookscorp.com>
```

[3] The underscore character is not strictly mail-safe, but it is used because the use of any other character to indicate a SPACE would seriously hamper readability.

An encoded word can be used in the *Your Email* section, but not in the address part between the < and the >. RFC 2047 specifies precisely where encoded words can be used with reference to the syntax of RFC 2822.

15.4 Post Office Protocol (POP)

The Post Office Protocol, version 3, is a standard protocol with STD number 53. Its status is elective, and it is described in RFC 1939. The older Post Office Protocol version 2, defined in RFC 0937, is a historic protocol with a status of not recommended.

The Post Office Protocol is an electronic mail protocol with both client (sender/receiver) and server (storage) functions. POP3 supports basic functions (download and delete) for electronic mail retrieval. More advanced functions are supported by IMAP4 (see 15.5, “Internet Message Access Protocol (IMAP4)” on page 591).

15.4.1 Connection states

After a POP3 client establishes a TCP connection to the server (using well-known port 110), the interaction between the client and server passes through three distinct states:

1. First, the POP3 server sends a greeting message to the client. Following this, the session then enters the *authentication state*. During this state, the client must authenticate itself to the server. This can be done using one of three methods:
   - USER/PASS: The combined use of a user ID and password (defined in RFC 1939)
   - APOP: Used to specify a name and an MD5 digest (also defined in RFC 1939)
   - AUTH: Used to specify a mechanism (such as TLS) by which both authentication and data protection can be provided (defined in RFC 1734)
2. If the server successfully authenticates the client, the session enters the *transaction state* in which the client can access the mailbox. During this period, the client can issue the commands listed in “Transaction state:” on page 590.

Archived

3. After the client sends the QUIT command, the session enters the *update state*. During this state, the server enacts all of the changes requested by the client's commands and then close the connection. If the connection is closed, for any reason, before a QUIT command is issued, none of the client's commands will take effect.

15.4.2 POP3 commands and responses

POP3 commands consist of a keyword and possibly one or more arguments following the keyword. Keywords are three or four characters long, and are separated from the arguments by a space. Each argument can be up to 40 characters long.

The server must send a response to each command issued by the client. This response can be up to 512 characters, and must begin with a status indicator signifying if the reply is positive or negative. These indicators are *+OK* or *-ERR*, and must be sent in uppercase.

As noted previously, POP3 interactions exist in three states. Commands can be issued from the authorization and transaction states, but not from the update state. With the exception of the QUIT command (which can be executed in both the authorization and transaction state), each command can only be executed in one of the states. Valid POP3 command, listed by state, are as follows:

- Authorization state:
  - USER name: User name for authentication.
  - PASS password: Password for authentication.
  - APOP name digest: The name and MD5 digest to be used for authentication (RFC 1939 indicates that implementation of this command is optional).
  - AUTH mechanism: The authentication/encryption mechanism to be used (RFC 1734 indicates that implementation of this command is optional).
  - QUIT: Terminate the authentication process.
- Transaction state:
  - STAT: Retrieve the number of messages and total size of the messages.
  - LIST [msg#]: If no *msg* number is provided, retrieve information about each message present in the mailbox.

    If a msg number is specified, the server returns information for that message.
  - RETR msg: Retrieve message number msg.
  - DELE msg: Delete message number msg.

- NOOP: Do nothing. The server returns a positive response.
- RSET: Cancel any previous delete commands.
- QUIT: Update the mailbox (delete any messages requested previously) and then end the TCP connection.

15.5 Internet Message Access Protocol (IMAP4)

The Internet Message Access Protocol, Version 4 is an electronic messaging protocol with both client and server functions. It is defined by RFC 3501. Similar to POP, IMAP4 servers store messages for multiple users to be retrieved upon client requests, but the IMAP4 model provides more functionality to users than does the POP model. IMAP4 allows clients to have multiple remote mailboxes from which messages can be retrieved, and allows users to choose any of those at any point. IMAP4 clients can also specify criteria for downloading messages, such as not transferring large messages over slow links. Additionally, IMAP4 always keeps messages on the server and replicates copies to the clients.

Another difference between POP and IMAP4 implementations is in the operational mode. Using POP, a client must always be connected to the server for changes to be made. However, IMAP4 allows clients to make changes both when connected and when disconnected. When disconnected, (referred to as a *disconnected client*), changes made on the client take effect on the server by periodic re-synchronization of the client and server. Let us discuss the underlying electronic mail models of IMAP4 first in order to understand the IMAP4 functions clearly. These are described in detail in RFC 1733 – Distributed Electronic Mail Models In IMAP4.

15.5.1 Fundamental IMAP4 electronic mail models

As defined in RFC 1733, there are three fundamental models implemented by the IMAP4 client and server: offline, online, and disconnected.

The offline model is similar to POP3's implementation (see 15.4, "Post Office Protocol (POP)" on page 589). An IMAP4 client connects to a server, downloads mail messages, and then disconnects from the server. Downloaded messages are then deleted from the server-based mailbox and exist only on the client's system.

The online model is the opposite of the offline model. In the online model, an IMAP4 client does not download messages from the server. Instead, it establishes a connection with the server, and then manipulates the mail while it remains in the server-based mailbox.

The disconnected model is a combination of both the offline and online models. In this model, an IMAP4 client connects to a server, downloads some or all of the messages, and disconnects. However, following the disconnect, the server does *not* delete the messages as it does in the offline model. The client can then manipulate the messages on the local system and later reconnect to the server. Upon reconnecting, the client's changes are synchronized with the server's mailbox, enacting any of the changes made while offline. In this model, the server remains the authoritative repository for the messages.

Each of these models have advantages and disadvantages, but because IMAP4 supports all these models, the client is able to switch to another model to meet whatever needs might exist at the time. Some of these advantages and disadvantages are listed in Table 15-5.

Table 15-5 Advantages and disadvantages of the IMAP4 models

Feature	Offline	Online	Disc
Use of multiple clients simultaneously	No	Yes	Yes
Minimum use of server connection time	Yes	No	Yes
Minimum use of server resources	Yes	No	No
Minimum use of client resources	No	Yes	No
Multiple remote mailboxes	No	Yes	Yes
Fast startup	No	Yes	No
Mail processing when not online	Yes	No	Yes

15.5.2 IMAP4 states

Similar to POP3 (see 15.4.1, "Connection states" on page 589), the IMAP4 session exists in different states. Some commands are valid for certain states and some of the commands are valid for all states. If the client sends a command that is not appropriate for that state, the server responds with an error message. The four states are:

Non-authenticated state In this state, the client has not yet authenticated with the server.

Authenticated state In this state, the client has identified itself to the server, and must select a mailbox to proceed.

Selected state In this state, a mailbox has been successfully selected, and actions can be taken against mail within the mailbox.

Logout state In this state, the connection has been ended either at the request of the client or for any other reason.

These states are illustrated in Figure 15-9.

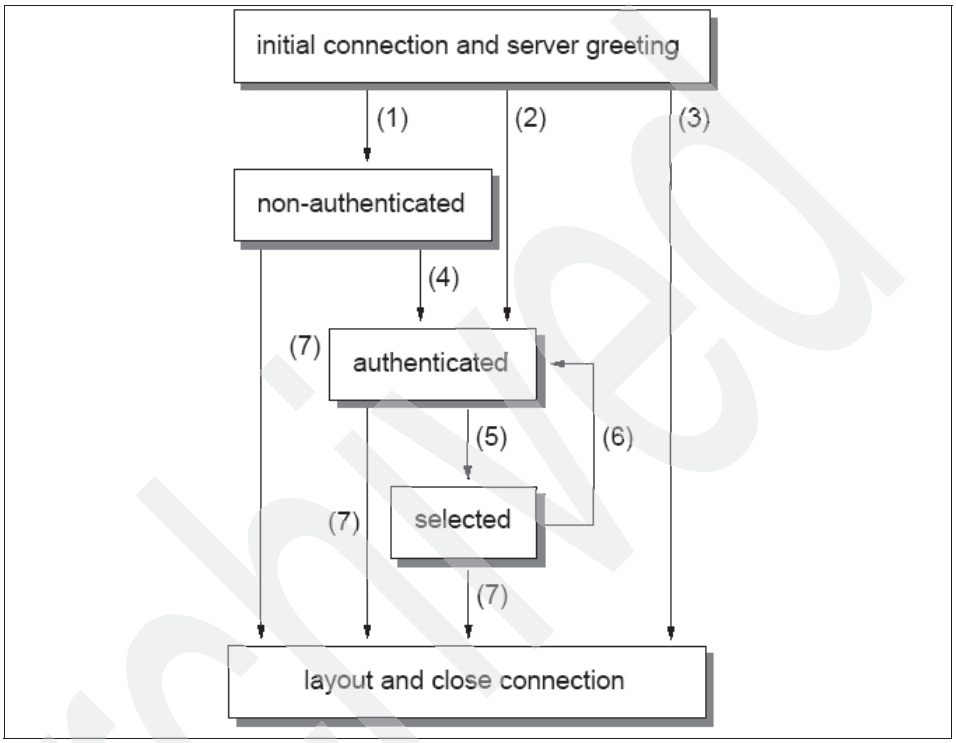


Figure 15-9 IMAP4 connection states

Where:

(1) Connection without pre-authentication (OK greeting)

(2) Pre-authenticated connection (PREAUTH greeting)

(3) Rejected connection (BYE greeting)

(4) Successful LOGIN or AUTHENTICATE command

(5) Successful SELECT or EXAMINE command

(6) CLOSE command, or failed SELECT or EXAMINE command

(7) LOGOUT command, server shutdown, or connection closed

15.5.3 IMAP4 commands and response interaction

IMAP4 clients establish a TCP connection to the server using well-known port 143. When the connection is established, the server sends a greeting message, after which the client and the server exchange data interactively. Whenever the client sends a command, the server sends a completion result response to this command. The server can also send data for any other reason. All commands and responses are in the form of lines ending with the <CRLF> sequence.

Although the server must respond to all client commands, it might not respond to them in the order in which they were received. In order to correlate the responses with the commands, the client commands begin with an identifier called a *tag*. For example, assume a client sends two commands to the server. The first has the tag *ABC005*, and the second with the tag *ABC006*. If it takes less time to process the second command, the server will respond to this command first, even though it was received second. In order to let the client know the response is for the second command, the response will include the ABC006 tag.

Client commands

Most of the IMAP4 commands must be used in the correct corresponding state (we define the states in 15.5.2, "IMAP4 states" on page 592), though some of them can be used in more than one state. The following list shows the commands and the states in which they are used:

- In any state:
  - CAPABILITY: Request a list of functions supported by the server.
  - NOOP: Do nothing. This is typically to reset an inactivity autologout timer on the server.
  - LOGOUT: Disconnect from the server.
- In the non-authenticated state:
  - AUTHENTICATE *mechanism*: This command requests a special authentication mechanism with an argument from the server. If the server does not support that mechanism, the server sends an error message. Valid mechanisms, defined in RFC 1731, include:
    - KERBEROS_V4
    - GSSAPI
    - SKEY
  - LOGIN *user pass*: This command sends the user name and password (in plain text).
  - STARTTLS: Begin TLS negotiation. Note that using TLS with IMAP4 is defined in RFC 2595.

- In the authenticated state:
  - SELECT *name*: Select the mailbox named *name*.
  - EXAMINE *name*: Select the mailbox named name, but in read-only mode.
  - CREATE *name*: Create a mailbox named name.
  - DELETE *name*: Delete the mailbox named name.
  - RENAME *oldName newName:* Change the name of the mailbox named *oldName* to *newName*.
  - SUBSCRIBE *name*: Add the mailbox named name to the subscription list.
  - UNSUBSCRIBE *name:* Remove the mailbox named name from the subscription list.
  - LIST *name mailbox*: Return a list of all names conforming to the name string within mailbox. If the name argument is not specified, all available names are listed.
  - LSUB *name*: Return a list of all mailboxes on the subscription list that conform to name. If the name argument is not specified, all mailboxes on the subscription list are returned.
  - STATUS *name item*: Return the status of item for the mailbox named name.
  - APPEND *mailbox message:* Appends the message text to the given mailbox as a new message. In the original RFC 3501 definition, only one message could be APPENDed at a time. However, a MULTIAPPEND extension was provided in RFC 3502, allowing multiple messages to be APPENDed at once. It was later extended by RFC 4469 to include a CONCATENATE option. This option enables a user to APPEND a message without first having to FETCH it from the server.
- In selected state:
  - CHECK: Request a checkpoint of the currently selected mailbox. The checkpoint is implementation-specific, but typically consists of "house-keeping" functions such as synchronizing a mailbox between a client and server.
  - CLOSE: Close the currently selected mailbox. This permanently removes all messages from the mailbox that were previously marked as deleted and returns the client to the authenticated state.
  - UNSELECT: Close the current mailbox without removing messages previously marked deleted. Note that this command is not defined in the original RFC 3501 specifications. Its implementation is defined in RFC 3691 and it is optional.

- EXPUNGE: Permanently removes all messages from the currently selected mailbox that were previously marked as deleted. This does not close the currently selected mailbox, nor does it remove the client from the selected state.
- SEARCH *criteria*: Search the mailbox for messages that match the specified criteria.
- FETCH *item message*: Retrieve the specified item associated with a message. Item can be a single thing, or a list of things.
- STORE *item message:* Store the specified item with the associated message in the mailbox.
- COPY *message mailbox:* Copies the specified message to the end of the specified destination mailbox.
- UID *name arguments:* Returns the unique identifier instead of message sequence numbers. This command is used with other commands.

> **Note:** Many of the IMAP commands, including CREATE, RENAME, FETCH, STORE, SEARCH, and APPEND have been extended by RFC 4466 to use the Augmented Backus-Naur Form (ABNF) syntax. This does not change the structure of the command, but only the manner in which the command is represented.

Server responses

The IMAP4 server's responses exist in three forms:

- Status
- Server data
- Command continuation requests

Depending on the message, these responses might or might not be tagged.

Status responses

Valid status response include:

OK	This response provides the client with information. If tagged, this indicates that a client command has completed successfully.
NO	This response indicates that an operational error has occurred on the server. If tagged, it indicates that a client command did not complete successfully.
BAD	This response provides an error message from the server. If tagged, the response is reporting a protocol-level error within a client's command.

PREAUTH This response is one of three possible greetings sent at connection startup. It is always untagged.

BYE This response indicates that the server is preparing to close the connection, and can be a part of the normal logout sequence, a panic shutdown, or an inactivity logout. It is always untagged.

Server data responses

Server data responses are unsolicited messages sent from the server to the client. They are always untagged, marked only by an asterisk. These messages can be sent at any time, including when there are outstanding client commands. Therefore, the client must be ready to accept server data responses at all times, even when it is awaiting a response indicating the completion of a command.

Server data responses indicate things such as the receipt of new mail, a change of state for a mailbox or message, or any other information for which the server needs to alert the client.

Command continuation request responses

There are cases in which a client's command is sent in two messages. In these cases, the client sends the first message, and then waits for a response from the server. When the server is ready for the second message, it sends the command continuation response. This response is marked by a plus (+) tag. An example of this is the logon sequence. In the following example, client commands are marked with a *C* and server responses are marked with an *S*:

```
C: A001 LOGIN
S: + Ready for additional command text
C: user1
S: + Ready for additional command text
C: password
S: A001 OK LOGIN completed
```

15.5.4 IMAP4 messages

There are two methods used to identify the messages: the unique identifier and the message sequence number. Some of the more common attributes are shown in the following sections. Refer to RFC 3501 for details.

Unique identifier (UID) message attribute

Every message has a 32-bit identifier, which, when it is combined with a unique identifier validity value, forms a 64-bit value. When a new message is added to the mailbox, a higher UID than those added previously is assigned to that message. Unique identifiers do not have to be contiguous, and also persist into

other sessions. This allows a client to access a message using the same information in every session.

Each mailbox has a unique identifier validity value. If it is not possible to use the same value for the next session, the new value must be greater than the value that was used in the previous session. For example, if a mailbox is deleted in one session and a new one created with the same name in the next session, the client might not realize that this is a new mailbox, because the mailbox name is the same. In this case, the unique identifier validity value must be changed. The unique identifier validity value is sent with the mailbox selection to the client as UIDVALIDITY.

Message sequence number message attribute

The message sequence number shows the relative position of the message in the mailbox, and must be in ascending order. The message sequence number is subject to change during the session, or between sessions. If a new message is added to a mailbox, it is given the next sequential number. If a message is deleted, the message numbers of messages remaining in the mailbox are recalculated.

Flags message attribute

Flags are used to show the current status of a message. These flags include:

\Seen	Message has been read.
\Answered	Message has been answered.
\Flagged	Message is marked for special attention.
\Deleted	Message is deleted for later permanent removal.
\Draft	Message has been completed.
\Recent	Message has arrived recently and this is the first session after its arrival. This flag cannot be changed by the client.

Note: IMAP4 does have the capability of using MIME (see 15.3, “Multipurpose Internet Mail Extensions (MIME)” on page 571). However, it is not restricted only to this. RFC 3516 offers an optional alternative method of handling binary content.

15.6 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 0934 - Proposed standard for message encapsulation (January 1985)
- RFC 0937 – Post Office Protocol – Version 2 (February 1985)
- RFC 1049 – A Content Type Header Field for Internet messages (March 1988)
- RFC 1652 – SMTP Service Extension for 8bit-MIMEtransport (July 1994)
- RFC 1731 – IMAP4 Authentication Mechanisms (December 1994)
- RFC 1733 – Distributed Electronic Mail Models in IMAP4 (December 1994)
- RFC 1870 – SMTP Service Extension for Message Size Declaration (November 1995)
- RFC 1939 – Post Office Protocol – Version 3 (May 1996)
- RFC 2045 – MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies (November 1996)
- RFC 2046 – MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types (November 1996)
- RFC 2047 – MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text (November 1996)
- RFC 2048 – MIME (Multipurpose Internet Mail Extensions) Part Four: Registration Procedures (November 1996)
- RFC 2049 – MIME (Multipurpose Internet Mail Extensions) Part Five: Conformance Criteria and Examples (November 1996)
- RFC 2156 – MIXER (MIME Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME (January 1998)
- RFC 2157 – Mapping between X.400 and RFC-822/MIME Message Bodies (January 1988)
- RFC 2183 – Communicating Presentation Information in Internet Messages (August 1997)
- RFC 2231 – MIME Parameter Value and Encoded Word Extensions (November 1997)
- RFC 2821 – Simple Mail Transfer Protocol (April 2001)
- RFC 2822 – Internet Message Format (April 2001)
- RFC 2554 – SMTP Service Extension for Authentication (March 1999)
- RFC 2595 – Using TLS with IMAP, POP3, and ACAP (June 1999)

Archive

- RFC 3030 – SMTP Service Extensions for Transmission of Large and Binary MIME Messages (August 1995)
- RFC 3207 – SMTP Service Extension for Secure SMTP over Transport Layer Security (February 2002)
- RFC 3461 – Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs) (January 2003)
- RFC 3501 – Internet Message Access Protocol – Version 4rev1 (March 2003)
- RFC 3502 – Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension (March 2003)
- RFC 3516 – IMAP4 Binary Content Extension (April 2003)
- RFC 3676 – The Text/Plain Format and DelSp Parameters (February 2004)
- RFC 3691 – Internet Message Access Protocol (IMAP) UNSELECT command (February 2004)
- RFC 3798 – Message Disposition Notification (May 2004)
- RFC 3974 – SMTP Operational Experience in Mixed IPv4/v6 Environments (January 2005)
- RFC 4466 – Collected Extensions to IMAP4 ABNF (April 2006)
- RFC 4469 – Internet Message Access Protocol (IMAP) CATENATE Extension (April 2006)

Archived

16

The Web

This chapter introduces some of the protocols and applications that have made the task of using the Internet both easier and very popular over the past years. In fact, World Wide Web traffic, which mostly uses the Hypertext Transfer Protocol (HTTP), greatly surpasses any other application protocol (such as Telnet and FTP) as using the most bandwidth across the Internet. Modern computer operating systems provide Web browser applications by default, some even provide Web servers, thus making it ever easier for users and businesses to explore and exploit the vast capabilities of worldwide networked computing.

The World Wide Web is a global hypertext system that was initially developed in 1989 by Tim Berners Lee at the European Laboratory for Particle Physics, CERN, in Switzerland to facilitate an easy way of sharing and editing research documents among a geographically dispersed group of scientists.

In 1993, the Web started to grow rapidly, which was mainly due to the National Center for Supercomputing Applications (NCSA) developing a Web browser program called Mosaic, an X Window System-based application. This application provided the first graphical user interface to the Web and made browsing more convenient. Today, there are Web browsers and servers available for nearly all platforms. The rapid growth in popularity of the Web is due to the flexible way people can navigate through worldwide resources in the Internet and retrieve them.

The number of Web servers is also increasing rapidly, and the traffic over port 80 on the NSF backbone has had a phenomenal rate of growth, too (see 4.1.1,

© Copyright IBM Corp. 1989-2006. All rights reserved.

"Ports" on page 144 for more information about ports). The NSFNET, however, was converted back to a private research network in 1995; therefore, comprehensive statistics of backbone traffic are not as easily available today.

The World Wide Web Consortium (W3C) is the entity responsible of development and maintenance of the standards of the Web.

Archived

16.1 Web browsers

Generally, a browser is referred to as an application that provides access to a Web server. Depending on the implementation, browser capabilities and thus structures vary. A Web browser, at a minimum, consists of an Hypertext Markup Language (HTML) interpreter and HTTP client that is used to retrieve HTML Web pages. Besides this basic requirement, many browsers also support FTP, NNTP, e-mail (POP and SMTP clients), among other features, with an easy-to-manage graphical interface. Figure 16-1 illustrates a basic Web browser structure.

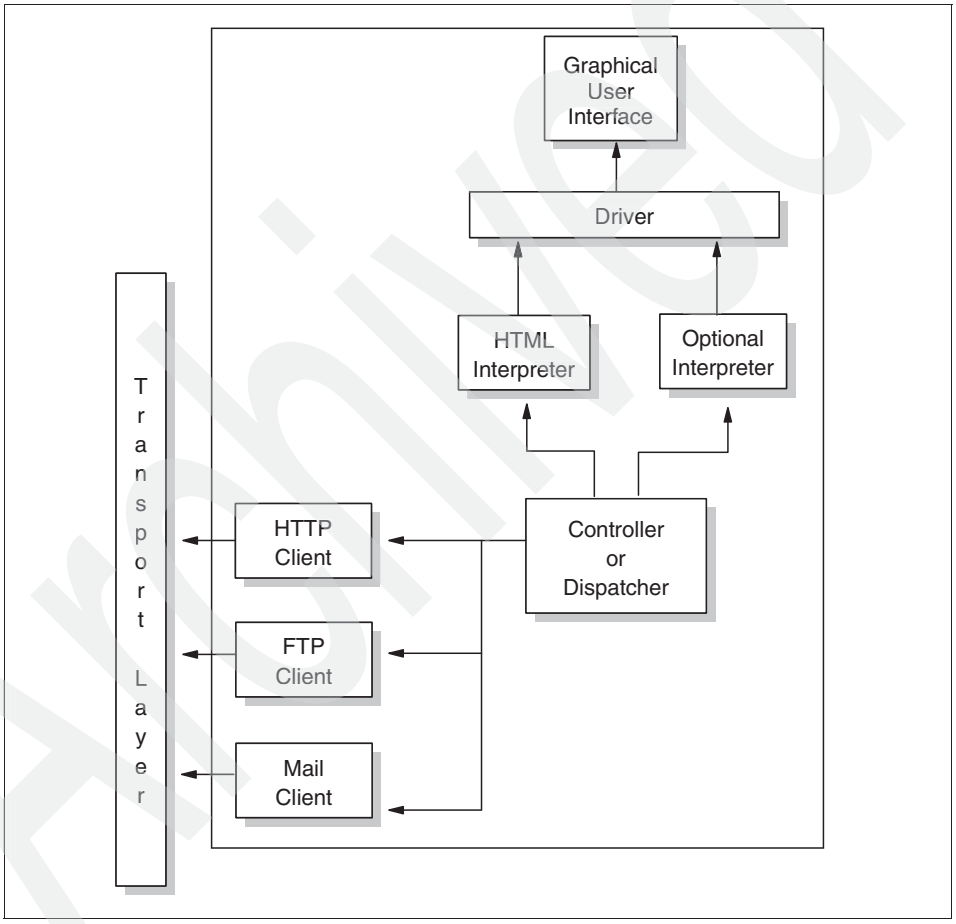


Figure 16-1 Structure of a Web browser

As with many other Internet facilities, the Web uses a client/server processing model. The Web browser is the client component. Examples of Web browsers include Mozilla Firefox, Netscape Navigator, and Microsoft Internet Explorer®.

Web browsers are responsible for formatting and displaying information, interacting with the user, and invoking external functions, such as Telnet, or external viewers for data types that Web browsers do not directly support. Web browsers have become the “universal client” for the GUI workstation environment, in much the same way that the ability to emulate popular terminals such as the DEC VT100 or IBM 3270 allows connectivity and access to character-based applications on a wide variety of computers. Web browsers are widely available for all popular GUI workstation platforms and are inexpensive.

16.2 Web servers

Web servers are responsible for servicing requests for information from Web browsers. The information can be a file retrieved from the server's local disk, or it can be generated by a program called by the server to perform a specific application function.

There are a number of public-domain Web servers available for a variety of platforms, including most UNIX variants, as well as personal computer environments such as Microsoft Windows. Some well-known public domain servers are CERN, NCSA httpd, and Apache servers.

IBM HTTP Server (current version is 6.1) is based on the Apache HTTP Server, which is the most popular server on the Web. This HTTP Server runs on IBM AIX 5L, Sun Solaris™, Microsoft Windows NT/2000, HP-UX, and Linux versions. IBM has enhanced the Apache-powered HTTP Server. For example, IBM has added SSL for secure transactions and offers full support, when part of the IBM WebSphere® bundle. IBM HTTP Server features include:

- Easy installation
- Support for SSL and TLS secure connections
- Fast Response Cache Accelerator
- IBM support as part of the WebSphere bundle
- Hardware crypto support
- Administration server that helps to administer and configure IBM HTTP Servers
- Help information that uses the easy-to-navigate design that is common to all WebSphere products

For more information, see:

`http://www.ibm.com/software/webservers/httpservers/sysreq/index.html`

16.3 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol is a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents. HTML is a tag language used to create hypertext documents. Hypertext documents include links to other documents that contain additional information about the highlighted term or subject. Such documents can contain other elements apart from text, such as graphic images, audio and video clips, Java applets, and even virtual reality worlds (which are described in VRML, a scripting language for that kind of elements). See “Hypertext Markup Language (HTML)” on page 615 for more information about HTML.

Nowadays, both HTTP 1.0 and HTTP 1.1 are stable specifications; therefore, W3C has closed the HTTP activity after its goal of creating a stable and weakness-free HTTP standard has been achieved.

16.3.1 Overview of HTTP

HTTP is based on request-response activity. A client, running an application called a browser, establishes a connection with a server and sends a request to the server in the form of a request method. The server responds with a status line, including the message's protocol version and a success or error code, followed by a message containing server information, entity information, and possible body content.

An HTTP transaction is divided into four steps:

1. The browser opens a connection.
2. The browser sends a request to the server.
3. The server sends a response to the browser.
4. The connection is closed.

On the Internet, HTTP communication generally takes place over TCP connections. The default port is TCP 80, but other ports can be used. This does not preclude HTTP from being implemented on top of any other protocol on the Internet or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

Except for experimental applications, current practice requires that the connection be established by the client prior to each request and closed by the server after sending the response. Both clients and servers should be aware that either party can close the connection prematurely, due to user action, automated timeout, or program failure, and should handle such closing in a predictable and

Archived

desirable fashion. In any case, the closing of the connection by either or both parties always terminates the current request, regardless of its status.

In simple terms, HTTP is a stateless protocol because it does not keep track of the connections. To load a page including two graphics, for example, a graphic-enabled browser will open three TCP connections: one for the page and two for the graphics. Most browsers, however, are able to handle several of these connections simultaneously.

This behavior can be rather resource-intensive if one page consists of a lot of elements, as quite a number of Web pages do. HTTP 1.1, as defined in RFC 2616, alleviates this problem to the extent that one TCP connection will be established per type of element on a page, and all elements of that kind will be transferred over the same connection respectively. These deviates from HTTP 1.0 by making the connections persistent.

However, if a request depends on the information exchanged during a previous connection, this information has to be kept outside the protocol. One way of tracking such persistent information is the use of cookies. A cookie is a set of information that is exchanged between a client Web browser and a Web server during an HTTP transaction. The maximum size of a cookie is 4 KB. All these pieces of information, or cookies, are then stored in one single file and placed in the directory of the Web browser. If cookies are disabled, that file is automatically deleted. A cookie can be retrieved and checked by the server at any subsequent connection. Because cookies are regarded as a potential privacy exposure, a Web browser should allow the user to decide whether or not he or she will accept cookies from a particular server. While cookies merely serve the purpose of keeping some kind of state for HTTP connections, secure client and server authentication is provided by the Secure Sockets Layer (SSL), which we describe in 22.7, “Secure Sockets Layer (SSL)” on page 854.

16.3.2 HTTP operation

In most cases, the HTTP communication is initiated by the user agent requesting a resource on the origin server. In the simplest case, the connection is established through a single connection between the user agent and the origin server, as shown in Figure 16-2 on page 607.

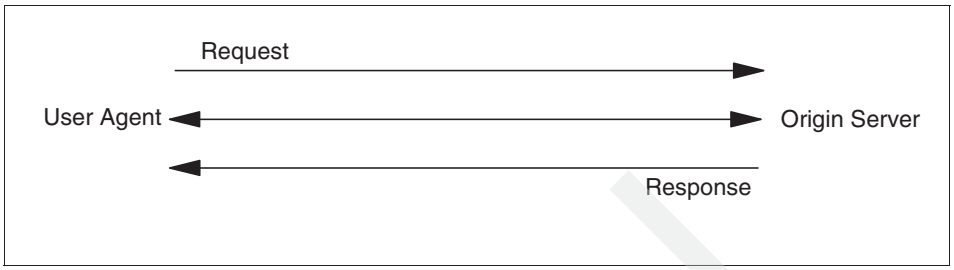


Figure 16-2 HTTP: Single client/server connection

In some cases, there is no direct connection between the user agent and the origin server. There is one (or more) intermediary between the user agent and origin server, such as a proxy, gateway, or tunnel. Requests and responses are evaluated by the intermediaries and forwarded to the destination or another intermediary in the request-response chain, as shown in Figure 16-3.

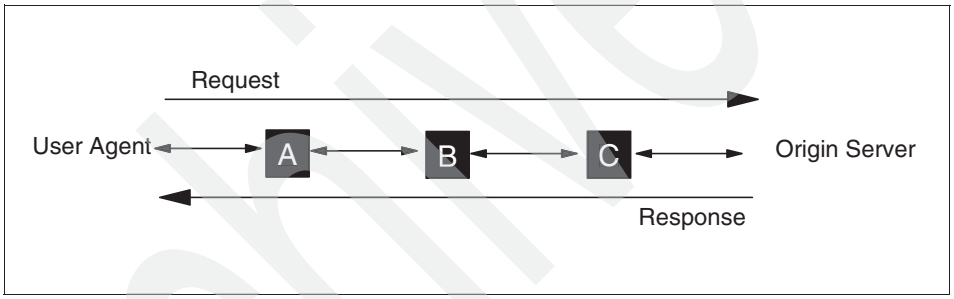


Figure 16-3 HTTP: Client/server connection with intermediaries in between

As described in “Application-level gateway (proxy)” on page 798, a proxy can handle the content of the data and therefore modify the data accordingly. When a request comes to a proxy, it rewrites all or part of the message and forwards the message to the next destination. A gateway receives the message and sends the message to the underlying protocols with an appropriate format. A tunnel does not deal with the content of the message; therefore, it simply forwards the message as it is.

Proxies, and gateways in general, can handle the caching of HTTP messages. This can dramatically reduce the response time and IP traffic in the network. Because tunnels cannot understand the message content, they cannot store cached data of HTTP messages. In the previous figure (Figure 16-3), if one of the intermediaries (A, B, and C) employs an internal cache for HTTP messages, the user agent can get a response from the intermediary if it is previously cached from the origin server in the response chain. Figure 16-4 on page 608 illustrates that A has a cached copy of an earlier response from the origin server in the

response chain. Therefore, if the server response for the request is not already cached in the user agent's internal cache, it can directly be obtained from A.

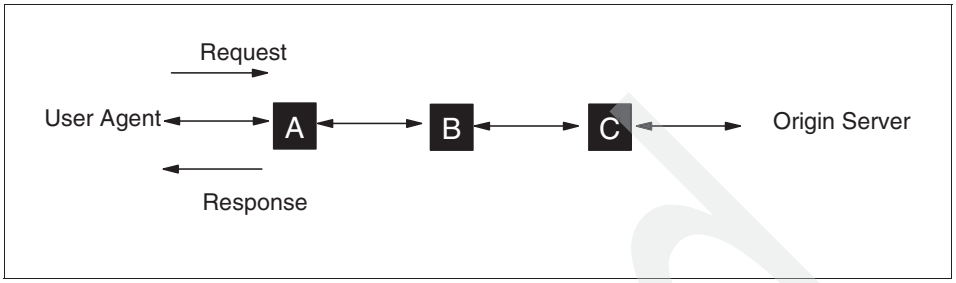


Figure 16-4 HTTP: Cached server response

Caching is not applicable to all server responses. Caching behavior can be modified by special requests to determine which server responses can or cannot be cached. For this purpose, server responses can be marked as non-cachable, public, or private (cannot be cached in a public cache). We discuss cache behavior and cachable responses in “HTTP caching” on page 614.

Protocol parameters

We provide some of the HTTP protocol parameters here. Refer to RFC 2616 for the full list and details.

- HTTP version

  HTTP uses a <major>.<minor> numbering scheme to indicate the versions of the protocol. The furthermost connection is performed according to the protocol versioning policy. The <major> number is incremented when there are significant changes in protocol, such as changing a message format. The <minor> number is incremented when the changes do not affect the message format.

  The version of HTTP messages is sent by an HTTP-Version field in the first line of the message. The HTTP-Version field is in the following format (refer to RFC 2822 for augmented Backus-Naur Form):

  ```
  HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
  ```

- Uniform Resource Identifiers (URIs)

  Uniform Resource Identifiers are generally referred to as WWW addresses and a combination of Uniform Resource Locators (URLs) and Uniform Resource Names (URNs). In fact, URIs are strings that indicate the location and name of the source on the server. See RFC 2616 and RFC 3986 for more details about the URI and URL syntax.

- HTTP URL

  The HTTP URL scheme enables you to locate network resources through the HTTP protocol. It is based on the URI Generic Syntax and described in RFC 3986. The general syntax of a URL scheme is:

  ```
  HTTP_URL = "http" "//" host [ ":" port ] [ abs_path ]
  ```

  The port number is optional. If it is not specified, the default value is 80.

HTTP message

HTTP messages consist of the following fields:

- Message types

  A HTTP message can be either a client request or a server response. The following string indicates the HTTP message type:

  ```
  HTTP-message = Request | Response
  ```

- Message header

  The HTTP message header field can be one of the following:

  - General header
  - Request header
  - Response header
  - Entity header

- Message body

  Message body can be referred to as entity body if there is no transfer coding has been applied. Message body simply carries the entity body of the relevant request or response.

- Message length

  Message length indicates the length of the message body if it is included. The message length is determined according to the criteria that is described in RFC 2616 in detail.

- General header field

  General header fields can apply both request and response messages. Currently defined general header field options are as follows:

  - Cache-Control
  - Connection
  - Date
  - Pragma
  - Transfer-Encoding

Archived

– Upgrade
– Via

Request

A request message from a client to a server includes the method to be applied to the resource, the identifier of the source, and the protocol version in use. A request message field is as follows:

```
Request = Request-Line
        *( general-header | request-header | entity-header )
        CRLF
        [ message-body ]
```

Refer to RFC 2616 for detailed information.

Response

An HTTP server returns a response after evaluating the client request. A response message field is as follows:

```
Request = Request-Line
        *( general-header | request-header | entity-header )
        CRLF
        [ message-body ]
```

Refer to RFC 2616 for detailed information.

Entity

Either the client or server might send Entity in the request message or the response message, unless otherwise indicated. Entity consists of the following:

- Entity header fields
- Entity body

Persistent connections

A significant difference between HTTP 1.1 and earlier versions of HTTP is that HTTP 1.1 uses a persistent connection as the default. In earlier version implementations, a separate TCP connection is established for each URL and clients have to make multiple requests for images and associated data on the same URL. This approach was causing congestion and performance problems in the network. Persistent HTTP connections have a number of advantages, most notably the reduction in TCP connections and subsequently, of waiting times.

Method definitions

Currently defined methods are as follows:

- Safe and idempotent methods

  Methods considered not to cause side effects are referred to as *safe*. Idempotent methods are GET, HEAD, PUT, and DELETE.

- OPTIONS

  This method allows the client to determine the options or requirements associated with a source or capabilities of a server without any resource retrieval.

- GET

  This method allows the client to retrieve the data that was determined by the request URI.

- HEAD

  This method allows the client to retrieve meta information about the entity that does not require you to transfer the entity body.

- POST

  The post function is determined by the server.

- PUT

  This method is similar to the post method with one important difference: The URI in post request identifies the resource that will handle enclosed entity.

- DELETE

  This methods requests that the server delete the source determined by the request URI.

- TRACE

  This method allows the client to see how the message was retrieved at the other side for testing and diagnostic purposes.

Status code definitions

The status code definitions are as follows:

- Informational (1xx)

  Informational status codes indicate a provisional response. Currently defined codes are as follows:

  – 100 Continue

  – 101 Switching Protocols

Archived

- Successful (2xx)

  This class of codes indicates that a particular request was successfully received, understood, and accepted. Currently defined codes are as follows:

  - 200 OK
  - 201 Created
  - 202 Accepted
  - 203 Non-Authoritative Information
  - 204 No Content -205 Reset Content
  - 206 Partial Content

- Redirection (3xx)

  This class of codes indicates that an action is required from the user agent in order to complete the request. Currently defined codes are as follows:

  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Moved Temporarily
  - 303 See Other
  - 304 Not Modified
  - 305 Use Proxy

- Client error (4xx)

  This class of codes indicates client errors. Currently defined codes are as follows:

  - 400 Bad Request
  - 401 Unauthorized
  - 402 Payment Required
  - 403 Forbidden -404 Not Found
  - 405 Method Not Allowed
  - 406 Not Acceptable
  - 407 Proxy Authentication Required
  - 408 Request Timeout
  - 409 Conflict
  - 410 Gone
  - 411 Length Required
  - 412 Precondition Failed

Archived

- 413 Request Entity Too Large
- 414 Request-URI Too Long
- 415 Unsupported Media Type

- Server error (5xx)

  This class of codes indicate client errors. Currently defined codes are as follows:

  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Service Unavailable
  - 504 Gateway Timeout
  - 505 HTTP Version Not Supported

Access authentication

HTTP provides an authentication mechanism to allow servers to define access permissions on resources and clients to use these resources. The authentication method can be one of the following:

- Basic authentication scheme

  Basic authentication is based on user IDs and passwords. In this authentication scheme, the server permits the connection only if the user ID and password are validated. In basic authentication, user IDs and passwords are not encrypted. They are encoded in Base64 format (see “Base64 encoding” on page 584). Therefore, the use of SSL or TLS is highly recommended.

- Digest authentication scheme

  Digest authentication scheme is an extension to HTTP and described in RFC 2617. In this authentication scheme, the user ID and a digest containing a hash value of the password are sent to the server. The server computes a similar digest and grants access to the protected resources if the two digests are equal. Notice that if the digest authentication is enabled, what is sent over the network is not simply an encrypted form of the password, which could be decrypted if one had the correct key, but is a one-hash value of the password, which cannot be decrypted. So, digest authentication provides a higher level of security than the Base64 encoded password. Unfortunately, digest authentication is not yet supported by all browsers.

Archived

Content negotiation

In order to find the best handling for different types of data, the correct representation for a particular entity body should be negotiated.

There are three types of negotiation:

- Server-driven negotiation

  The representation for a response is determined according to the algorithms located at the server.

- Agent-driven negotiation

  The representation for a response is determined according to the algorithms located.

- Transparent negotiation

  This is a combination of both server-driven and agent-driven negotiation. It is accomplished by a cache that includes a list of all available representations.

HTTP caching

One of the most important features of HTTP is caching capability. Because HTTP is a distributed information-based protocol, caching can improve the performance significantly. There are a number of functions that come with the HTTP 1.1 protocol to use caching efficiently and properly.

In most cases, client requests and server responses can be stored in a cache within a reasonable amount of time to handle the corresponding future requests. If the response is in the cache and is accurate, there is no need to request another response from the server. This approach not only reduces the network bandwidth requirement, but also increases the speed. There is a mechanism that the server estimates a minimum time in which the response message will be valid. That means, an expiration time is determined by the server for that particular response message. Therefore, within this time, the message can be used without referring to the server.

Consider that this time is exceeded and there is a need for that response message. The data inside the message might have been changed (or not) after the expiration date. To be able to ensure whether the data is changed or not, a validation mechanism is defined as follows:

- Expiration mechanism

  In order to decide whether the data is fresh or not, an expiration time is determined. In most cases, the origin server explicitly defines the expiration time for a particular response message within that message. If this is the case, the cached data can be used to send from cache for subsequent requests within the expiration time.

If the origin server did not define any expiration time, there are some methods to estimate/calculate a reasonable expiration time (such as the Last-Modified time). Because this is not originated from the server, use this cautiously.

- Validation mechanism

  When the expiration time is exceeded, there is a possibility that the data is stale. In order to ensure the validation of the response message, the cache has to check with the origin server (or possibly an intermediate cache with a fresh response) whether the response message is still usable. HTTP 1.1 provides conditional methods for this purpose.

  When an origin server sends a full response, it attaches some sort of validator to the message. This then is used as a *cache validator* by the user agent or the proxy cache. The client (user agent or the proxy cache) generates a conditional request with a cache validator attached to it. The server then evaluates the message and responds with a special code (usually 304, Not Modified) and no entity body. Otherwise, the server sends the full response (including the entity body). This approach avoids an extra round-trip if the validator does not match and also avoids sending the full response if the validator matches.

Refer to RFC 2616 for more details about HTTP caching.

16.4 Content

A Web server can serve static or dynamic (generated by a program upon invocation) content. This section discusses some commonly used technologies used to provide content and to facilitate interaction between a Web server and an application server that is not typically directly accessible to a client (for example, a Web browser).

16.4.1 Static content

Static content, or static pages, usually consist of data associated with a URL that does not change very often and does not depend on any client input. That is not to say that static content does not change at all, because a Web page may be updated frequently. This content is usually in the form of some markup language, such as HTML or XML.

Hypertext Markup Language (HTML)

HTML is one of the major attractions of the Web. It has an architected set of tags understood by all Web browsers and Web servers, although as new features are added to HTML, they might not be supported by older Web browsers. These tags are device independent. The same document can be sent from a personal

computer, an AIX 5L or UNIX machine, or a mainframe, and the Web browser on any client machine can understand the HTML tags and build the data stream to display it on the target device. HTML tags describe basic elements of a Web document, such as headers, paragraphs, text styles, and lists. There are also more sophisticated tags to create tables and to include interactive elements, such as forms, scripts, or Java applets.

After document writers and programmers have mastered HTML, those skills are applicable to any operating system on any machine, provided that it has a Web browser.

Because HTML supports hypertext, it allows document writers to include links to other HTML documents. Those documents might be on the same machine as the original, or they might be on a machine on another network on the other side of the world; such is the power of HTML links.

Extensible Markup Language (XML)

Extensible Markup Language (XML) describes a class of data objects called XML documents that are stored on computers, and partially describes the behavior of programs that process these objects. XML is an application profile, or restricted form, of SGML. The goal of XML is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

16.4.2 Client-side dynamic content

The extension of functionality into the client has led to the development of technologies to leverage dynamic content on the client's side.

Programs and applets

When a Java program is started from inside an HTML (Web) page, it is called a Java applet, as opposed to a Java program, which is executed from the command line or otherwise on the local system. Applets are downloaded by the Web browser from a server and, by definition, are somewhat limited in the way they can use resources of the local system.

Originally, Java applets were not supposed to touch anything local (outside of its Java virtual machine, or JVM™), and could only communicate back to the server from which it was downloaded. With Java 1.1, applets can be signed with security keys and certificates and can therefore be authenticated. Therefore, an applet can be authorized to access local resources, such as file systems, and it can communicate with other systems.

JavaScript

JavaScript™ is an HTML extension and programming language, developed by Netscape, which is a simple object-based language compatible with Java. JavaScript programs are embedded as a source directly in an HTML document. They can control the behavior of forms, buttons, and text elements. It is used to create dynamic behavior in elements of the Web page. In addition, it can be used to create forms whose fields have built-in error checking routines.

16.4.3 Server-side dynamic content

The complement of client-side dynamic content generation is, of course, server-side content generation. By executing the necessary functions of the server's side, this technology can leverage a server's processing power and its ability to coordinate requests with objects.

Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is a means of allowing a Web server to execute a program that is provided by the Web server administrator, rather than retrieving a file. CGI programs allow a Web server to generate a dynamic response, usually based on the client's input. A number of popular Web servers support the CGI, and a variety of programming languages can be used to develop programs that interface with CGI. However, CGI programs are not easily portable across platforms, unless using Perl.

Server-specific APIs

Some Web servers offer specific APIs that allow developers to create programs that can be invoked for special purposes on certain events. Those APIs are usually quite powerful, but offer no portability across Web server platforms. The most popular server-specific APIs are Netscape Server API (NSAPI) and Microsoft Internet Information Server API (ISAPI).

Servlets

In order to spare resources on clients and networks, Java applets can be executed on the server rather than downloaded and started at the client. Such programs are then referred to as servlets. Though that method requires a significantly more powerful server, it is highly suitable for environments with medialess systems, such as network computers. This method is usually very portable across platforms and incurs little processing cost.

Server-side includes (SSI)

This is a technology that a Java-enabled Web server (meaning, a Web server with a servlet engine) can use to convert a section of an HTML file into an

alternative dynamic portion each time the document is sent to the client's browser. This dynamic portion invokes an appropriate servlet and passes to it the parameters it needs. The replacement is performed at the server and it is completely transparent to the client. Pages that use this technology have the extension .shtml instead of .html (or .htm).

JavaServer Pages (JSP)

This is an easy-to-use solution for generating HTML (or other markup languages such as XML) pages with dynamic content. A JSP™ file contains combinations of HTML tags, NCSA tags (special tags that were the first method of implementing server-side includes), <SERVLET> tags, and JSP syntax. JSP files have the extension .jsp. One of the many advantages of JSP is that it enables programmers to effectively separate the HTML coding from the business logic in Web pages. JSP can be used to access reusable components, such as servlets, JavaBeans™ (reusable Java objects), and Java-based Web applications. JSP also supports embedding inline Java code within Web pages. JSPs are typically compiled into servlets for execution.

Objects

One main thrust in developing content on the Web is the use of objects. In general, objects allow for decreased application development cost and effort by promoting the reusability of code. In addition, they allow for cooperation and coordination between different processes (and machines) by enabling operations that change the state of particular objects. That is, the term object is sometimes used to describe an implementation of reusable data structures and functions, but can also be used to described the instantiation of those data structures and functions.

JavaBeans

According to its inventors at JavaSoft™, a JavaBean is a reusable software component that can be manipulated visually by using a builder tool. The JavaSoft definition allows for a broad range of components that can be thought of as beans.

JavaBeans can be visual components, such as buttons or entry fields, or even an entire spreadsheet application. JavaBeans can also be non-visual components, encapsulating business tasks or entities, such as processing employee paychecks, a bank account, or even an entire credit rating component. Non-visual beans still have a visual representation, such as an icon or name, to allow visual manipulation. While this visual representation might not appear to the user of an application, non-visual beans are depicted on screen so that developers can work with them.

JavaBeans can only be manipulated and reused if they are built in a standardized way. To build beans, JavaSoft provides the JavaBeans API, an architecture that defines a software component model for Java. The JavaBeans architecture delivers four key benefits:

- Support for a range of component granularity, because beans can come in different shapes and sizes.
- Portability, because the API is platform neutral. A bean, especially non-visual components developed under Windows, for example, behaves the same whether it is run under Windows, AIX 5L, Solaris, or even z/OS, OS/400®.
- Uniform, high-quality API. Ideally, every platform that supports Java will support the entire JavaBeans API.
- Simplicity. The API is simple, universal, compact, and easy to learn and begin to use.

The JavaBeans API defines the distinguishing characteristics of a bean, specifically, how they look and feel.

CORBA

Common Object Request Broker Architecture (CORBA) is a standard for distributed applications in which computers remotely invoke methods on objects residing on other computers. CORBA allows interconnection of objects and application regardless of language, location, or computer architecture.

An application is enabled to use distributed objects by using the Object Request Broker (ORB). The ORB transparently forwards remote object requests (for example, a method invocation) to the appropriate server objects, dispatches the requests, and returns the results (for example, a method return value).

The Internet Inter-ORB Protocol (IIOP) is a communications protocol based on the CORBA specifications provided by the Object Management Group (OMG). For further information about the Object Management Group, refer to the OMG home page at the following URL:

`http://www.omg.org`

Enterprise JavaBeans

Enterprise JavaBeans™ (EJB™) is a Java component that can be combined with other EJBs and other Java components to create a distributed, multi-tiered application. Figure 16-5 on page 620 shows the EJB environment. An EJB client, such as a servlet or Java application, interacts with the EJB server through Remote Method Invocation (RMI) to access the objects (EJBs), which it coordinates. The actual object data is contained in some data source.

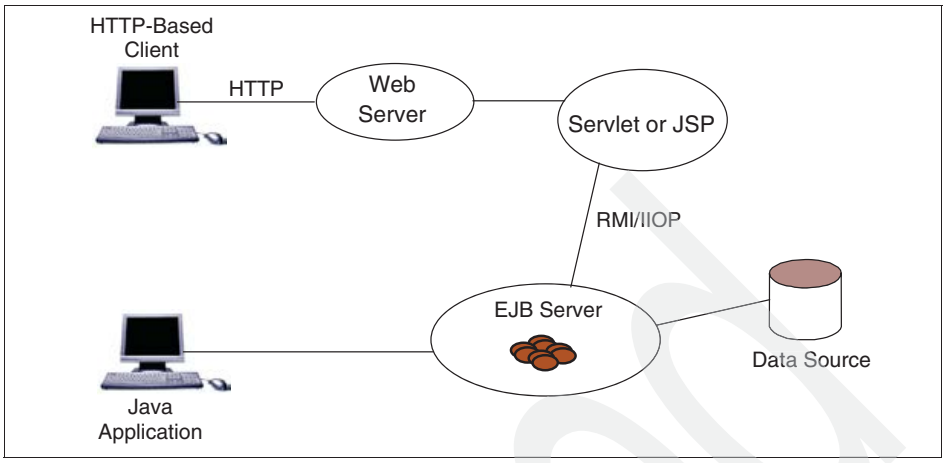


Figure 16-5 The Enterprise JavaBeans environment

Remote Method Invocation (RMI)

RMI is a standard protocol for communication between Java objects residing on different computers. RMI provides a way for client and server applications to invoke methods across a distributed network of clients and servers running the JVM. You can invoke methods on the remote RMI object as you do on a local Java object. RMI is cross-platform, but not cross-language (Java only).

Sun Microsystems, with IBM and others, developed a more portable version of RMI, which uses the Object Management Group's (OMG) Internet Inter-ORB Protocol. IIOP is necessary for Java 2 Platform, Enterprise Edition (J2EE™) deployments to be interoperable with CORBA systems.

Types of EJBs

There are two types of Enterprise JavaBeans (EJB):

- Entity EJBs

  They encapsulate permanent data, which is stored in a data source such as a database or a file system, and associated methods to manipulate that data. In most cases, an entity bean must be accessed in some transactional manner. Instances of an entity bean are unique and they can be accessed by multiple users. For example, information about a bank account can be encapsulated in an entity bean. An account entity bean might contain an account ID, an account type (checking or savings), a balance variable, and methods to manipulate these variables.

- Session EJBs

  They encapsulate ephemeral (nonpermanent) data associated with a particular EJB client. Unlike the data in an entity bean, the data in a session bean is not stored in a permanent data source and no harm is caused if this data is lost. However, a session bean can update data in an underlying database, usually by accessing an entity bean. A session bean can also participate in a transaction.

  When created, instances of a session bean are identical, though some session beans can store semipermanent data that make them unique at certain points in their life cycle. A session bean is always associated with a single client; attempts to make concurrent calls result in an exception being thrown.

  For example, the task associated with transferring funds between two bank accounts can be encapsulated in a session bean. Such a transfer session bean can find two instances of an account entity bean (by using the account IDs), and then subtract a specified amount from one account and add the same amount to the other account.

- Message-driven EJBs

  These beans where added in the EJB 2.0 specification. These beans are similar to a session bean, except they respond to a Java Message Service (JMS) service, thus allowing asynchronous message processing.

16.4.4 Developing content with IBM Web application servers

IBM offers numerous powerful solutions for Web application development and deployment. For more information about these and other IBM products, refer to the following link:

`http://www.software.ibm.com/webservers`

16.5 RFCs relevant to this chapter

The following RFCs provide detailed information about the connection protocols and architectures presented throughout this chapter:

- RFC 1945 – Hypertext Transfer Protocol -- HTTP/1.0 (May 1996)
- RFC 2616 – Hypertext Transfer Protocol -- HTTP/1.1 (June 1999)
- RFC 2617 – HTTP Authentication (June 1999)
- RFC 2817 – Upgrading to TLS Within HTTP/1.1 (May 2000)
- RFC 2822 – Internet Message Format (April 2001)
- RFC 3986 – Uniform Resource Identifiers (January 2005)

Archived

17

Network management

With the growth in size and complexity of the TCP/IP-based internetworks the need for network management became very important. To address this, in 1988 the Internet Architecture Board (IAB) issued RFC 1052 detailing its recommendation to achieve this management. The suggestion adopted two different approaches:

- The Simple Network Management Protocol (SNMP)
- ISO Common Management Information Services/Common Management Information Protocol (CMIS/CMIP)

The original plan detailed by RFC 1052 involved using SNMP as a short-term solution to the network management problem. The development of SNMP was to be kept simple, facilitating rapid deployment of the protocol throughout the Internet community. After the immediate management needs were met, albeit temporarily, by SNMP, thorough research and development could be performed on CMIS/CMIP. Ultimately, this protocol would then be deployed as a permanent solution, replacing SNMP.

However, this plan allowed SNMP to gain widespread usage throughout the Internet community. As a result, CMIS/CMIP was never fully deployed. In fact, only two RFCs describing CMIS/CMIP were released. RFC 1095, which described the CMIS protocol over TCP/IP (CMOT) was released in 1989. It was ultimately obsoleted by RFC 1189, released in 1990, which proposed the standard for CMOT and CMIP. The status of RFC 1189 is currently historic, and is not widely used in today's networks.

© Copyright IBM Corp. 1989-2006. All rights reserved.

Given these circumstances, this chapter discusses the SNMP model and implementation. Also included is a discussion on the NETSTAT utility. Though this command is not RFC defined, it is available on most platforms and is useful in both monitoring and managing local connections on a system.

17.1 The Simple Network Management Protocol (SNMP)

The fundamental use of the Simple Network Management Protocol (SNMP) is to manage all aspects of a network, as well as applications related to that network. To do this, SNMP has been architected to perform two services:

- Monitor: SNMP implementations allow network administrators to monitor their networks in order to--among other things--ensure the health of the network, forecast usage and capacity, and in problem determination. Aspects which can be monitored vary in granularity, and can be something as global as the total amount of IP traffic experienced on a single host, or can be as minute as the current status of a single TCP connection.

  Additionally, the SNMP architecture allows components of the SNMP model to notify network administrators should a problem occur on a network. For example, if a link were to break or an interface to deactivate for some reason, SNMP can send a message to alert network administrators that this has occurred.
- Manage: In addition to monitoring a network, SNMP provides the capability for network administrators to affect aspects with the network. Values which regulate network operation can be altered, allowing administrators to quickly respond to network problems, dynamically implement new network changes, and to perform real-time testing on how changes may affect their network.

SNMP implements a manager/agent/subagent model, which conforms very closely to the client/server model (11.1.1, “The client/server model” on page 408). RFC 1157 defines the components and interactions involved in an SNMP community, which include:

- A Management Information Base (MIB, discussed in 17.1.1, “The Management Information Base (MIB)” on page 625)
- An SNMP agent (discussed in 17.1.2, “The SNMP agent” on page 630)
- An manager (discussed in 17.1.3, “The SNMP manager” on page 631)
- SNMP subagents (discussed in 17.1.4, “The SNMP subagent” on page 632)

17.1.1 The Management Information Base (MIB)

The management information base defines a set of objects which can be monitored or managed using an SNMP implementation. The current MIB, MIB-II, is defined in RFC 1213 and replaces the MIB-I definition outlined in RFC 1156. It is updated by RFCs 4022, 4113, and 4293. MIB-II defines the groups of information which should be made available in any SNMP implementation in a TCP/IP based network. These groups are as shown in Table 17-1.

Table 17-1 Group names and descriptions available in an SNMP implementation

Group name	Description of objects within the group	RFC defining the group's MIB
System	Basic system information, such as the system's name, description, and how much time has passed since the last time the system was restarted.	RFC 3418
Interfaces	Information about network interfaces, including a list of interfaces, and statistics specific to these interfaces.	RFC 2863
IP	Information and statistics on IP traffic.	RFC 4293
ICMP	Statistics on ICMP input and output.	RFC 4293
TCP	General information about the TCP layer (such as timeout values and the total number of TCP connections) as well as information about specific TCP connections (such as the connection's state, addresses, and ports).	RFC 4022
UDP	General information about the UDP layer (such as the number of UDP packets sent and delivered) as well as information about specific UDP usage (such as addresses and ports).	RFC 4113
EGP	Statistics on external gateway protocol traffic.	RFC 1156 RFC 1213
Transmission	This group is not yet implemented, but was created as a placeholder for when Internet-standard definitions for managing transmission media emerge.	RFC 1213

Archive

Group name	Description of objects within the group	RFC defining the group's MIB
SNMP	Information and statistics related to the SNMP environment.	RFC 3411 RFC 3412 RFC 3413 RFC 3414 RFC 3415 RFC 3418

Note that these constitute only the minimum implementation. An implementation can also create it's own groups and objects which are application or platform-specific. In such cases, the MIB must be built into the SNMP agent itself, or be made available to the agent using the distributed programming interface (DPI, see 11.2.3, "The SNMP distributed programming interface (SNMP DPI)" on page 419). Additionally, each managed node supports only those groups that are appropriate. For example, if there is no gateway, the EGP group need not be supported. If a group is appropriate, all objects in that group must be supported.

Objects within a MIB are defined using the Structure of Management Information Version 2 (SMIv2), defined in RFC 2578.

Structure of Management Information Version 2 (SMIv2)

SMIv2 defines the rules for how managed objects are described and how management protocols can access these objects. The description of managed objects is made using a subset of the Abstract Syntax Notation One (ASN.1, ISO standard 8824), a data description language. The object type definition uses the following syntax:

```
objectName          OBJECT-TYPE
     SYNTAX          syntax
     UNITS           "units"
     MAX-ACCESS      access
     STATUS          status
     DESCRIPTION     "descriptiveText"
     REFERENCE       "referenceText"
     INDEX           {indexTypes}
     DEFVAL          {defaultValue}
     ::= { group # }
```

These fields are defined as follows:

OBJECT-TYPE	A textual name, called the *object descriptor*, for the object type along with its corresponding *object identifier* defined later.
SYNTAX	The abstract syntax for the object type. It can be a choice of SimpleSyntax (Integer, Octet String, Object Identifier, Null) or an ApplicationSyntax (NetworkAddress, Counter, Gauge, TimeTicks, Opaque) or other application-wide types (see RFC 2578 for more details).
UNITS	This field is optional and applies only when the object maintains a value that is unit specific. For example, an object that monitors time might specify UNIT “seconds” or UNIT “minutes”.
MAX-ACCESS	Defines the access that a user has to this object. Valid values include accessible-for-notify, read-only, read-write, read_create, and not-accessible.
STATUS	Defines the status of the object. Valid values include current, deprecated, and obsolete.
DESCRIPTION	A textual description of the semantics of the object type.
REFERENCE	This field is optional and can be used to include a text string referencing some other document.
INDEX	This field is optional and is only used when the object is part of a conceptual row of objects.
DEFVAL	This field is optional and is only used when the object should have a default value.
group	The RFC 1213-defined group to which the object belongs.
#	The object’s position within the RFC 1213-defined group.

Archive

As an example, consider the object ifType, defined by RFC 2863. ifType is part of the Interface group defined by RFC 1213. Its SMIv2 definitions is illustrated in Figure 17-1.

```
ifType    OBJECT-TYPE
     SYNTAX         IANAifType
     MAX-ACCESS     read-only
     STATUS         current
     DESCRIPTION
          "The type of interface. Additional values for ifType are
          assigned by the Internet Assigned Numbers Authority
          (IANA), through updating the syntax of the IANAifType
          textual convention."
     ::= { ifEntry 3 }
```

Figure 17-1 SMIv2 definition of sysUpTime

Object identifiers (OIDs)

A managed object not only has to be described but identified, too. This is done using the ASN.1 object identifier (OID). The object identifier reserves a set of numbers for different groups. Each object is identified by a string of numbers indicating the hierarchy to which it belongs. Referring back to the example of ifType, this object has an OID of 1.3.6.1.2.1.2.2.1.3. This can initially be broken into two parts:

ifEntry 1.3.6.1.2.1.2.2.1

ifType 3

Note that the terms ifType, 1.3.6.1.2.1.2.2.1.3, and ifEntry.3 are functionally interchangeable. However, ifType's OID can be further broken down as follows:

ifTable 1.3.6.1.2.1.2.2

ifEntry 1

ifType 3

Again, the terms ifType, 1.3.6.1.2.1.2.2.1.3, ifTable.1.3, and ifEntry.3 are all functionally interchangeable. The OID can continue to be broken down because each digit has a specific meaning. The significance of each digit adheres to the following rules:

- The first digit defines the node administrator:
  - 1 for ISO

Archived

  - 2 for CCITT
  - 3 for the joint ISO-CCITT
- The possible values for the second digit are determined by the first digit. In this case, the ISO node administrator defines *3* for use by other organizations.
- The third digit's potential values again depend on the first and second digits. But if the first two digits are 1.3, *6* is defined for the use of the U.S. Department of Defense.
- In the fourth group, the Department of Defense has not indicated how it will manage its group, so the Internet community assumed *1* for its own.
- The fifth group was approved by IAB to be:
  - 1 for the use of OSI directory in the Internet
  - 2 for object identification for management purposes
  - 3 for object identification for experimental purposes
  - 4 for object identification for private use

Archived

This if further illustrated in Figure 17-2, which shows a mapping of how the OID number for ifType is determined.

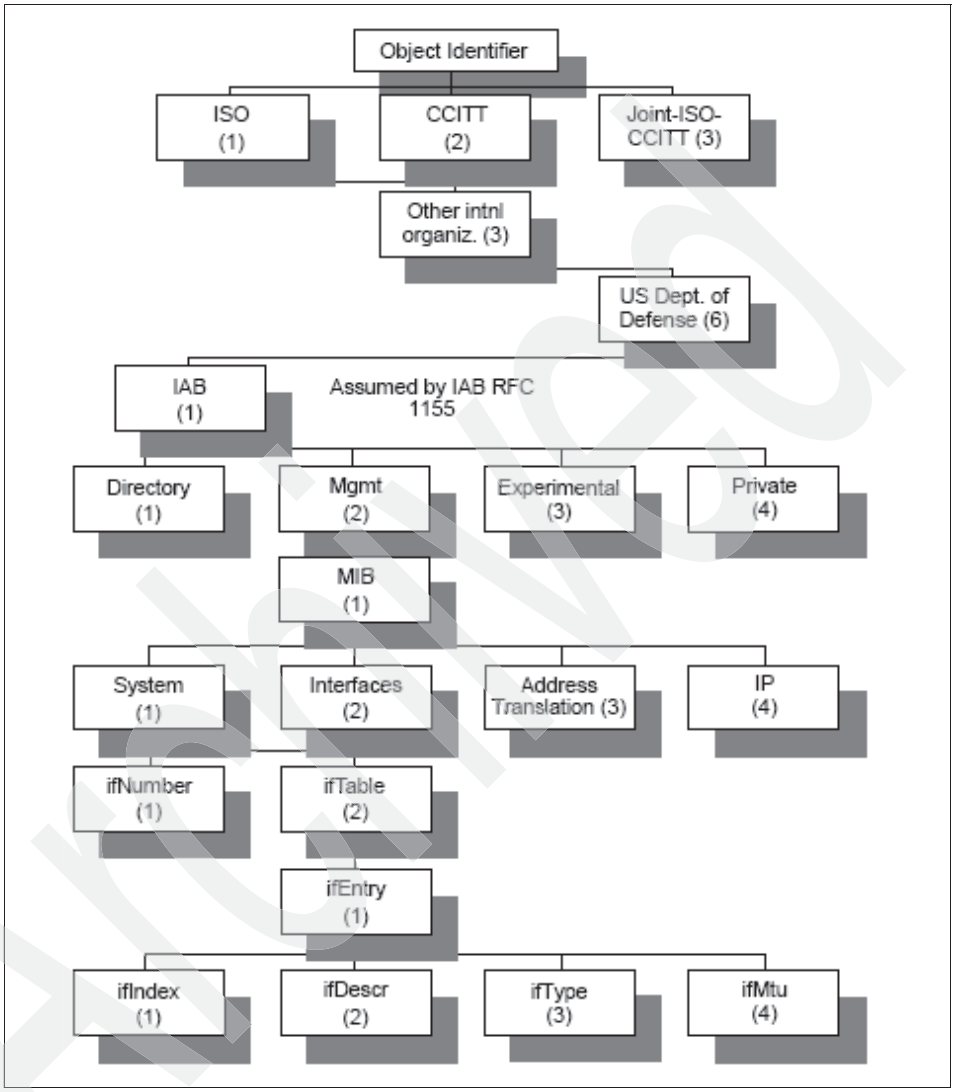


Figure 17-2 Finding ifType's OID number

17.1.2 The SNMP agent

The SNMP agent acts as a server in the client/server model and listens on well-known UDP port 161 for requests from SNMP managers (we discuss managers in 17.1.3, "The SNMP manager" on page 631). Additionally, the SNMP

agent is typically responsible for supporting both the system and the SNMP groups defined by RFC 1213.

Upon receiving a request from a manager, the agent determines if the requesting management station has the authority to access the SNMP community. If so, the agent obtains the value of the requested objects and returns them to the manager.

Finally, the agent can accept connections from subagents (discussed in the next section) to make MIBs available to SNMP managers. Upon receiving a request from a manager for an object supported by a subagent, the agent forwards the request to the appropriate subagent. This is illustrated later in Figure 17-5 on page 636.

17.1.3 The SNMP manager

The SNMP manager, also referred to as a Network Management Station (NMS), provides a user interface through which network administrators can monitor and manage their network. The manager fulfills the role of a client in the client/server model and is available in a variety of formats including command-line interfaces, graphical user interface (GUI) applications, and fully automated applications.

The SNMP manager is responsible for issuing requests to the SNMP agent. These requests can be queries to obtain the value of a MIB object, or they can be requests to set the value of a MIB object. SNMP managers also can listen for notifications or alerts, called traps, generated by components in the SNMP community. For additional information about traps, see 17.1.6, “SNMP traps” on page 638.

Note: Communication between an SNMP manager and the SNMP agent occurs using the communication structure outlined by the Simple Network Management Protocol. SNMP usually employs UDP as a transport.

An SNMP manager can make the following types of requests to the SNMP agent:

`getRequest` Requests that the agent return the value of the specified object.

`getNextRequest` Requests that the agent return the first valid value following the specified object. For example, assume a **`getNext`** is executed for ifType (1.3.6.1.2.1.2.2.1.3). Assuming the first valid instance of ifType is ifType.1 (1.3.6.1.2.1.2.2.1.3.1), this is the value that the SNMP agent will return.

getBulkRequest Performs the same function as the **get** request, but allows the manager to query more than one object per request. This is only valid using the SNMPv2c security model (see "The GetBulkRequest" on page 642).

setRequest Requests that the SNMP agent set the value of the specified object.

walk Implements a series of **getNext** requests such that an entire sequence of objects is returned to the manager. In each iteration of the **getNext** series, the last object returned becomes the next object on which a **getNext** is executed. The **walk** ends when an object is returned that is beyond the scope of the request. An example of this is provided in Figure 17-6 on page 637.

Note that the **walk** request is not architected in the SNMP communication that occurs between the SNMP manager and SNMP agent. Instead, it is a convention widely used by most SNMP managers.

17.1.4 The SNMP subagent

An SNMP subagent supports its own MIB, which might be an RFC-architected MIB, or might be a proprietary (referred to as enterprise-specific) MIB. For example, a TCP/IP subagent would most likely support the IP, ICMP, TCP, UDP, and Interface groups defined in RFC 1213. However, an individual software company might want use SNMP to make available information specific to their software. To do this, they can create a subagent that supports their enterprise-specific MIB.

The subagent, upon initializing, opens a DPI connection to the SNMP agent. This occurs by first querying the agent, as though the subagent were a manager, for information about the agent's DPI ports. Note that this information is maintained by the SNMP agent in the following two objects:

dpiPortForTCP.0 1.3.6.1.4.1.2.2.1.1.1.0

dpiPortForUDP.0 1.3.6.1.4.1.2.2.1.1.2.0

The agent's response directs the subagent to the correct port over which a DPI connection can be opened. With this information, the subagent can interact with the agent, as described in 11.2.3, "The SNMP distributed programming interface (SNMP DPI)" on page 419.

17.1.5 The SNMP model

The interaction between SNMP components is agent-restrictive: A manager can communicate only with an agent, and a subagent can communicate only with an agent. In no aspects of the model will a subagent communicate with a manager. If a manager needs to obtain or set the value of an object supported by a subagent, the request is delivered to the agent.

The agent, upon realizing that the request is for an object in a MIB other than one it supports, attempts to find the object in one of the MIBs registered by a subagent. Upon finding the correct MIB, the agent passes the request to the subagent.

The subagent then locates the correct value and passes it back to the SNMP agent. The agent then forwards this value back to the manager. This process, as well as the relationship that exists between manager, agents, and subagents, is illustrated in Figure 17-5 on page 636.

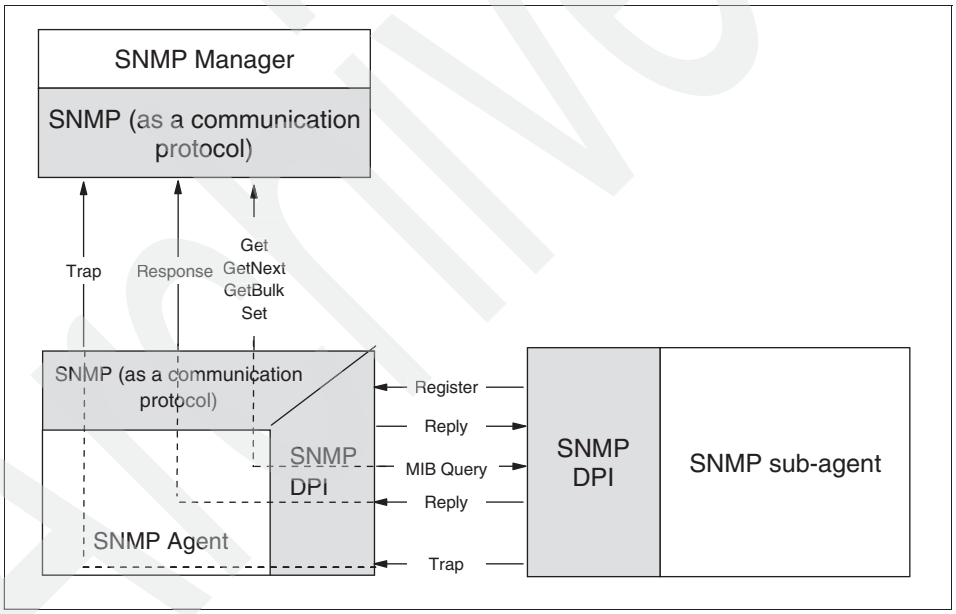


Figure 17-3 The SNMP model

Archive

Messages sent between SNMP agents and managers use a Protocol Data Unit (PDU) preceded by the SNMP header. The header specifies the version of SNMP being used, as well as authentication credentials. The PDU contains information regarding the type of request or response contained in the PDU, and in the case of a response, the actual value of the queried objects. The SNMP message format is illustrated in Figure 17-4.

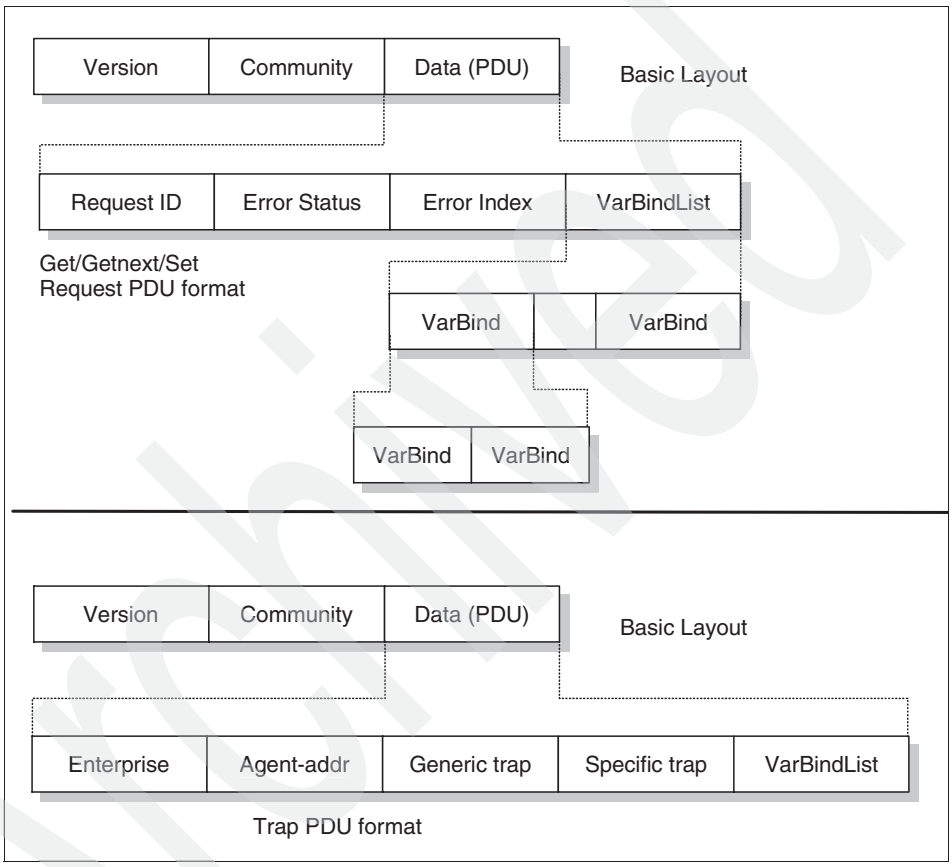


Figure 17-4 The SNMP message format

This format is defined in RFC 1157, and the fields are defined as follows:

Version Indicates the version of SNMP being used (valid values are 1, 2, and 3).

Community The SNMP community to which the request is directed, or from which the response originated.

Request ID Serializes request/response iterations. A manager uses this to determine which responses correspond to a specific request.

Error Status Specified on responses to inform the manager of any errors encountered with the request. On request PDUs, this value is set to 0. Valid values are:

- **noError** No error was encountered.
- **tooBig** The response was too large to deliver in a PDU.
- **noSuchName** The requested object does not exist.
- **badValue** The returned value does not adhere to ASN.1 encoding standards.
- **readOnly** A set request was executed for an object which is read-only.
- **genErr** An error not covered by the other types occurred.

Error Index An integer indicating which object in a list of objects caused the error reported by the errorStatus.

VarBind The list of objects requested and, for a response, their values.

Enterprise The entity (defined by the sysObjectID MIB object) generating an enterprise-specific trap.

Agent-addr The address of the SNMP agent from which the trap was generated.

Generic trap The generic trap type that indicates the type of trap sent. If this is an enterprise-specific trap, this is set to 6 (see 17.1.6, “SNMP traps” on page 638).

Specific trap The value of an enterprise-specific trap (see 17.1.6, “SNMP traps” on page 638).

Archived

To illustrate the process of querying an object using SNMP, refer back to the example of the ifType object. Figure 17-5 illustrates the sequence of events occurring when executing a **getNext** request for the ifType object.

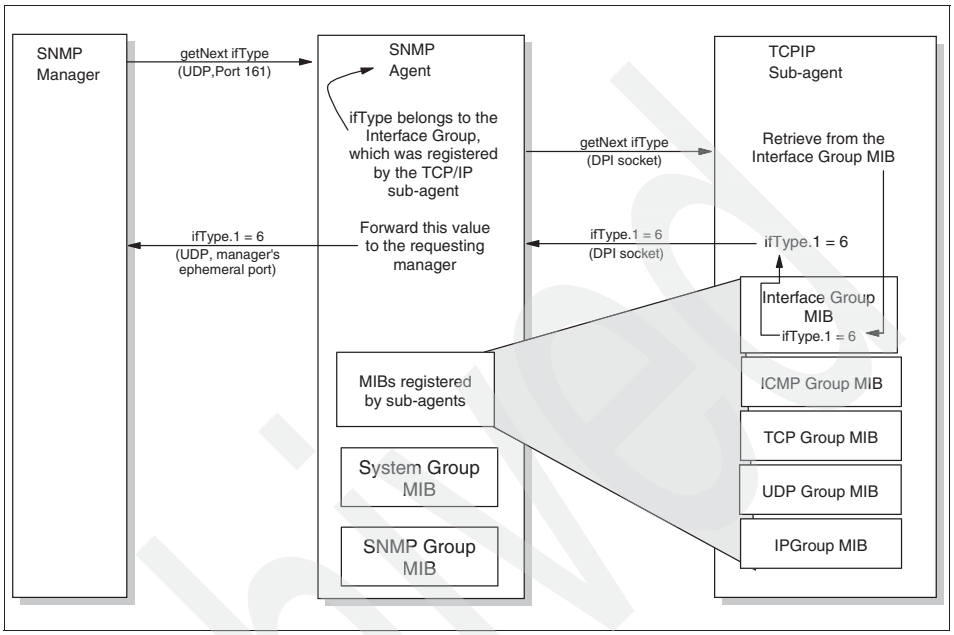


Figure 17-5 An example of SNMP manager, agent, and subagent interaction

The sequence of events is as follows:

1. In this example, a **getNext** request is executed by the SNMP manager for ifType.
2. The SNMP agent does not recognize ifType as an object from the System Group or SNMP Group MIBs. It does, however, recognize ifType as an object from a MIB registered by a subagent. In this case, the MIB is the Interface Group MIB, and was registered by the TCP/IP subagent.
3. The SNMP agent passes the **getNext** over the DPI socket to the TCP/IP subagent.
4. The subagent recognizes ifType as an object in the Interface MIB and obtains the next object with a valid value: ifType.1.
5. The subagent sends a response back to the agent with the object ifType.1 and the value 6 (which, by RFC definition, is Ethernet).
6. The agent then replies back to the manager, again indicating that the object is ifType.1 and the value is 6.

Archived

Note that, had the **getNext** request been for an object in the System Group or SNMP Group MIBs, the SNMP agent would have recognized the object and responded directly to the manager. No subagent would have been involved in this circumstance.

While the concept of the **get**, **getNext**, **getBulk**, and **set** processes are somewhat simple, the process of executing a **walk** warrants additional discussion. As noted previously, a **walk** request is never actually sent to the SNMP agent. Instead, the walk is a manager convention, and is implemented as a series of **getNext** requests. In each iteration of the series, the object returned previously by the agent becomes the next object specified on the **getNext** request. The first time the agent returns a value outside of the range specified on the **walk** request, the **walk** ends. To demonstrate this, assume that there are three instances of ifType at the time a **walk** is issued for the ifType object: ifType.1 through ifType.3. The processing of the walk proceeds as illustrated in Figure 17-6.

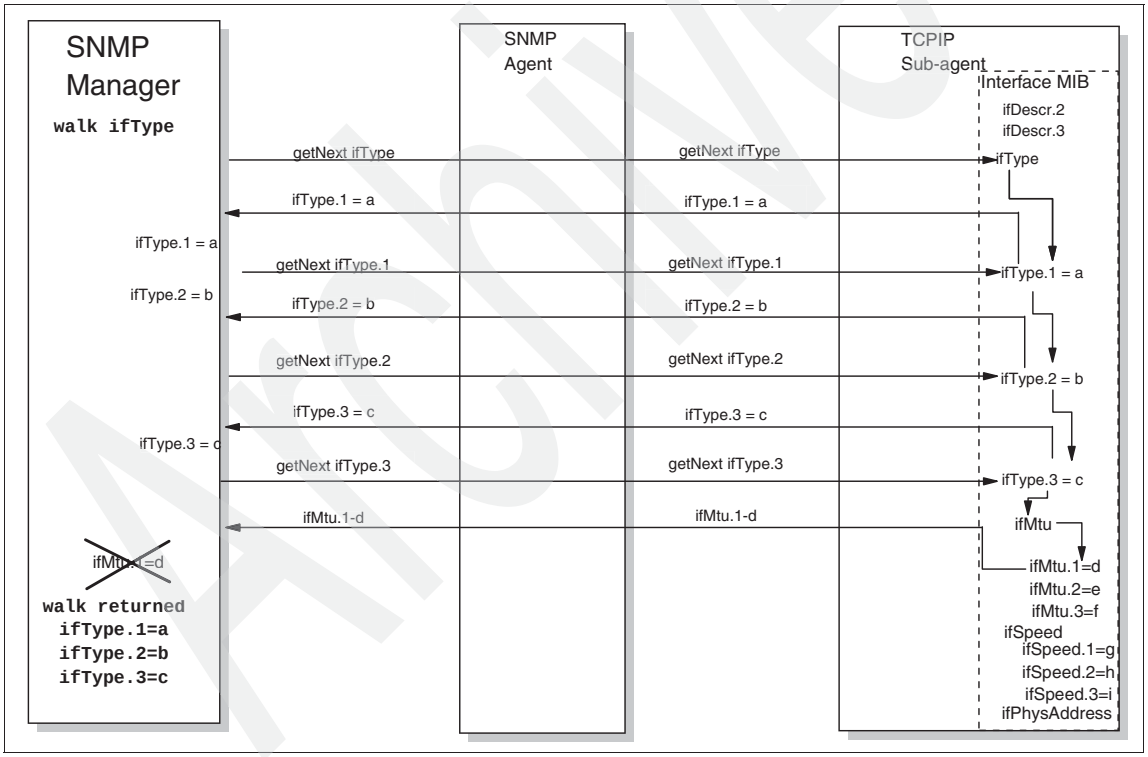


Figure 17-6 An example of a walk on ifType

In this illustration, we see the following progression:

1. The user executes a **walk** on ifType, which the manager implements by first executing a **getNext** on ifType. This is forwarded to the TCP/IP subagent by the SNMP agent.
2. The subagent obtains the next valid value following ifType, ifType.1, and returns this value to the agent. The agent passes this information back to the manager.
3. The manager takes the ifType.1 response and executes a **getNext** for it.
4. The subagent locates ifType.1 and returns the next valid value: ifType.2.
5. The manager executes a **getNext** for ifType.2
6. The subagent returns ifType.3
7. The manager executes a **getNext** for ifType.3
8. The subagent returns ifMtu.1
9. The manager recognizes that ifMtu.1 is outside the scope of ifType, and assumes that the **walk** is complete.

17.1.6 SNMP traps

Traps are asynchronous notifications of events occurring within an SNMP community. They can be generated both by SNMP agents and SNMP subagents. Additionally, they can be RFC architected (called a generic trap type) or they can be proprietary (called enterprise-specific). Architected traps, defined in RFC 1215, are as follows:

coldStartEvent Notifies managers that the SNMP agent is reinitializing and that the configuration might have been altered. This trap belongs to the RFC 1213-architected System group, and is therefore supported by the SNMP agent.

warmStartEvent Notifies managers that the SNMP agent is reinitializing, but there has been no alteration of the configuration. This trap belongs to the RFC 1213-architected System group, and is therefore supported by the SNMP agent.

linkDownEvent Notifies managers that an interface has been deactivated. Information identifying the interface is included in the trap. This trap belongs to the RFC 1213-architected Interface group and is usually supported by a TCP/IP specific subagent.

linkUpEvent Notifies managers that an interface has been activated. Information identifying the interface is included in the trap. This trap belongs to the RFC 1213-architected Interface group and is usually supported by a TCP/IP-specific subagent, or by an SNMP agent that manages its own TCP/IP MIBs.

snmpAuthenFailureEvent
Notifies managers that a user attempting to access the SNMP community did not provide the credentials needed to gain authorization by the SNMP agent. This trap belongs to the RFC 1213-architected SNMP group, and therefore is supported by the SNMP agent.

egpNeighborLossEvent
Notifies manages that a relationship with an EGP neighbor no longer exists. Information identifying the EGP neighbor is included in the trap. This trap belongs to the RFC 1213-architected EGP group, and therefore is usually supported by an EGP-specific subagent or a TCP/IP specific-subagent.

entSpecificEvent This trap is a placeholder that allows SNMP agent or subagent implementations to create enterprise-specific traps.

Traps generated by an SNMP agent are usually delivered to managers using well-known UDP port 162. However, SNMP implementations might provide a configuration option to allow traps to be sent to other user-determined ports. If a subagent generates a trap, the trap is not sent directly from the subagent to a manager. Instead, the trap is passed over the DPI connection to the agent, who then sends out the trap to the managers (see Figure 17-3 on page 633).

17.1.7 SNMP versions

There are three versions of SNMP available, usually referred as SNMPv1 (RFC 1157), SNMPv2 (RFC 1901), and SNMPv3 (RFC 3414). Additionally, RFC 3584 was created to specify how all three versions can coexist with a single SNMP community. The security functions provided by the SNMP protocols are categorized into the following two models:

- Community-based Security Model, whose data is protected by nothing more than a password, referred to as the community name (see Figure 17-4 on page 634). Community-based security allows the SNMP agent to authenticate a request based on the community name used and the IP address from which the request originated. This level of security is provided by the SNMPv1 and SNMPv2 Community-based Security Models.

- User-based Security Model (USM), which provides different levels of security, based on the user accessing the managed information. To support this security level, the SNMPv3 framework defines several security functions, such as USM for authentication and privacy and vIew-based Access Control Model (VACM, defined in RFC 3415), which provides the ability to limit access to different MIB objects on a per-user basis and the use of authentication and data encryption for privacy.

SNMPv1

Version 1 of SNMP incorporates the basics of SNMP already covered in this chapter. Therefore, there is no need for additional discussion except to note that SNMPv1 does not allow **`getBulk`** requests. Such requests executed in an SNMPv1 community, depending on the SNMP manager implementation, will result either in an error or in downgrading the request to a series of **`get`** requests.

SNMPv2

The framework of Version 2 of the Simple Network Management Protocol (SNMPv2) was published in April 1993 and consists of 12 RFCs, including the first, RFC 1441, which is an introduction. In August 1993, all 12 RFCs became a proposed standard with the status elective.

This framework consists of the following disciplines:

- Structure of Management Information (SMI)

  Definition of the OSI ASN.1 subset for creating MIB modules. See RFC 2578 for a description.

- Textual conventions

  Definition of the initial set of textual conventions available to all MIB modules. See RFC 2579 for a description.

- Protocol operations

  Definition of protocol operations with respect to the sending and receiving of PDUs. See RFC 3416 for a description.

- Transport mappings

  Definition of mapping SNMPv2 onto an initial set of transport domains because it can be used over a variety of protocol suites. The mapping onto UDP is the preferred mapping. The RFC also defines OSI, DDP, IPX, and so on. See RFC 3417 for a description.

- Protocol instrumentation

  Definition of the MIB for SNMPv2. See RFC 3418 for a description.

- Administrative framework

  Definition of the administrative infrastructure for SNMPv2, the User-based Security Model for SNMPv2 and the Community-based SNMPv2. See RFCs 1901, 1909, and 1910 for descriptions.

- Conformance statements

  Definition of the notation compliance or capability of agents. See RFC 2580 for a description.

The following sections describe the major differences and improvements from SNMPv1 to SNMPv2.

SNMPv2 entity

An SNMPv2 entity is an actual process that performs network management operations by generating or responding, or both, to SNMPv2 protocol messages by using the SNMPv2 protocol operations. All possible operations of an entity can be restricted to a subset of all possible operations that belong to a particular administratively defined party (refer to “SNMPv2 party”). An SNMPv2 entity can be member of multiple SNMPv2 parties. The following local databases are maintained by an SNMPv2 entity:

- One database for all parties known by the SNMPv2 entity that can be:
  - Operation realized locally
  - Operation realized by proxy interactions with remote parties or devices
  - Operation realized by other SNMPv2 entities
- Another database that represents all managed object resources that are known to that SNMPv2 entity.
- And at least a database that represents an access control policy that defines the access privileges accorded to known SNMPv2 parties.

An SNMPv2 entity can act as an SNMPv2 agent or manager.

SNMPv2 party

An SNMPv2 party is a conceptual, virtual execution environment whose operation is restricted, for security or other purposes, to an administratively defined subset of all possible operations of a particular SNMPv2 entity (refer to “SNMPv2 entity”). Architecturally, each SNMPv2 party consists of:

- A single, unique party identity.
- A logical network location at which the party executes, characterized by a transport protocol domain and transport addressing information.

- A single authentication protocol and associated parameters by which all protocol messages originated by the party are authenticated as to origin and integrity.
- A single privacy protocol and associated parameters by which all protocol messages received by the party are protected from disclosure.

The GetBulkRequest

The **GetBulkRequest** is defined in RFC 3416 and is thus part of the protocol operations. A **GetBulkRequest** is generated and transmitted as a request of an SNMPv2 application. The purpose of the **GetBulkRequest** is to request the transfer of a potentially large amount of data, including, but not limited to, the efficient and rapid retrieval of large tables. The **GetBulkRequest** is more efficient than the **GetNext** request in case of retrieval of large MIB object tables. The syntax of the **GetBulkRequest** is:

```
GetBulkRequest [ non-repeaters = N, max-repetitions = M ]
                ( RequestedObjectName1,
                  RequestedObjectName2,
                  RequestedObjectName3 )
```

Where:

RequestedObjectName1, 2, 3
: MIB object identifier, such as sysUpTime. The objects are in a lexicographically ordered list. Each object identifier has a binding to at least one variable. For example, the object identifier ipNetToMediaPhysAddress has a variable binding for each IP address in the ARP table and the content is the associated MAC address.

N
: Specifies the non-repeaters value, which means that you request only the contents of the variable next to the object specified in your request of the first N objects named between the parentheses. This is the same function as provided by the **GetNextRequest**.

M
: Specifies the max-repetitions value, which means that you request from the remaining (number of requested objects - N) objects the contents of the M variables next to your object specified in the request. Similar to an iterated **GetNextRequest** but transmitted in only one request.

With the **GetBulkRequest**, you can efficiently get the contents of the next variable or the next M variables in only one request.

Assume the following ARP table in a host that runs an SNMPv2 agent:

```
Interface-Number  Network-Address  Physical-Address  Type

       1            10.0.0.51     00:00:10:01:23:45  static
       1             9.2.3.4      00:00:10:54:32:10  dynamic
       2            10.0.0.15     00:00:10:98:76:54  dynamic
```

An SNMPv2 manager sends the following request to retrieve the sysUpTime and the complete ARP table:

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
                ( sysUpTime,
                  ipNetToMediaPhysAddress,
                  ipNetToMediaType )
```

The SNMPv2 agent responds with a response PDU:

```
Response (( sysUpTime.0 =  "123456" ),
          ( ipNetToMediaPhysAddress.1.9.2.3.4 =
                                    "000010543210" ),
          ( ipNetToMediaType.1.9.2.3.4 =  "dynamic" ),
          ( ipNetToMediaPhysAddress.1.10.0.0.51 =
                                     "000010012345" ),
          ( ipNetToMediaType.1.10.0.0.51 =  "static" ))
```

The SNMPv2 manager continues with:

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
                ( sysUpTime,
                  ipNetToMediaPhysAddress.1.10.0.0.51,
                  ipNetToMediaType.1.10.0.0.51 )
```

The SNMPv2 agent responds with:

```
Response (( sysUpTime.0 =  "123466" ),
          ( ipNetToMediaPhysAddress.2.10.0.0.15 =
                                    "000010987654" ),
          ( ipNetToMediaType.2.10.0.0.15 =
                                       "dynamic" ),
          ( ipNetToMediaNetAddress.1.9.2.3.4 =
                                       "9.2.3.4" ),
          ( ipRoutingDiscards.0 =  "2" ))
```

This response signals the end of the table to the SNMPv2 manager. Using the **getNextRequest**, this same result requires four iterations of queries.

Archived

InformRequest

An **InformRequest** is generated and transmitted as a request from an application in an SNMPv2 manager entity that wants to notify another application, acting also in an SNMPv2 manager entity, of information in the MIB view[1] of a party local to the sending application. The packet is used as an indicative assertion to the manager of another party about information accessible to the originating party (manager-to-manager communication across party boundaries). The first two variables in the variable binding list of an **InformRequest** are sysUpTime.0 and snmpEventID.i[2], respectively. Other variables can follow.

The new administrative model

It is the purpose of the administrative model for SNMPv2 to define how the administrative framework is applied to realize effective network management in a variety of configurations and environments.

The model entails the use of distinct identities for peers that exchange SNMPv2 messages. Therefore, it represents a departure from the community-based administrative model of the original SNMPv1. By unambiguously identifying the source and intended recipient of each SNMPv2 message, this new strategy improves on the historical community scheme both by supporting a more convenient access control model and allowing for effective use of asymmetric (public key) security protocols in the future. Figure 17-7 on page 645 illustrates the new message format.

[1] A MIB view is a subset of the set of all instances of all object types defined according to SMI.
[2] snmpEventID.i is an SNMPv2 manager-to-manager MIB object that shows the authoritative identification of an event.

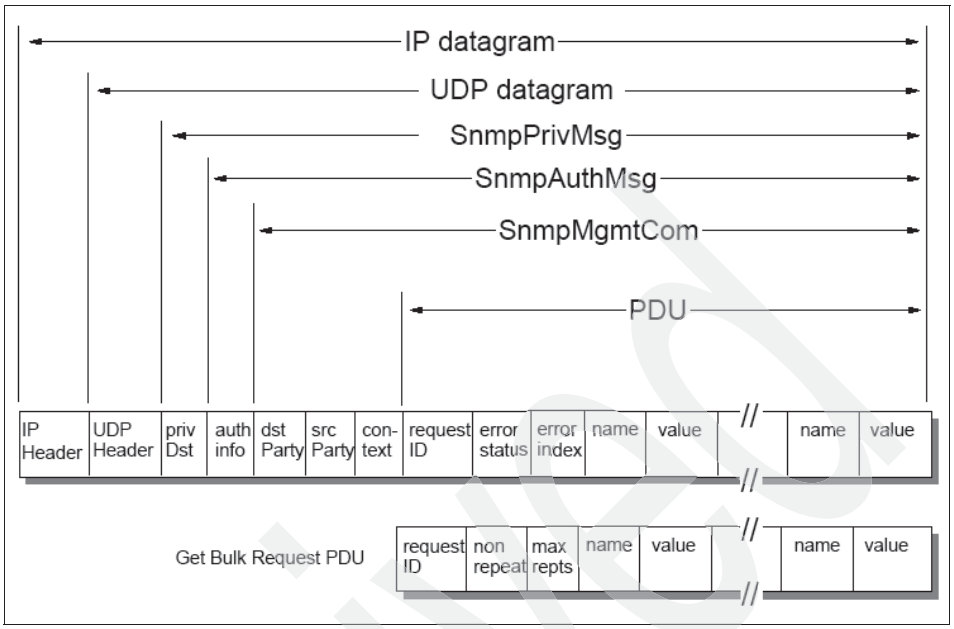


Figure 17-7 The SNMPv2 message format

In this figure, the fields are defined as follows:

PDU Includes one of the following protocol data units:

- GetNextRequest
- GetRequest
- Inform
- Report
- Response
- SNMPv2-Trap
- SetRequest

The **`GetBulkRequest`** has a different PDU format, as shown earlier (refer to “The GetBulkRequest” on page 642).

SnmpMgmtCom (SNMP Management Communication)
Adds the source party ID (srcParty), the destination party ID (dstParty), and the context to the PDU. The context specifies the SNMPv2 context containing the management information referenced by the communication.

SnmpAuthMsg This field is used as authentication information from the authentication protocol used by that party. The SnmpAuthMsg is serialized according to ASN.1 BER[3] and can then be encrypted.

SnmpPrivMsg SNMP Private Message
An SNMPv2 private message is an SNMPv2 authenticated management communication that is (possibly) protected from disclosure. A private destination (privDst) is added to address the destination party.

The message is then encapsulated in a normal UDP/IP datagram and sent to the destination across the network.

For further information, refer to the previously mentioned RFCs.

SNMPv3

SNMPv3 is described in RFCs 3410 through 3415. SNMPv3 is not a replacement of SNMPv1 or SNMPv2, but rather is an extension to the existing SNMP architecture.

SNMPv3 supports the following:

- A new SNMP message format
- Authentication for messages
- Security for messages
- Access control
- Continued support for SNMPv2

The User-based Security Model (USM), described in RFC 3414, specifies using MD5 and hashing algorithms. This provides data integrity, security, and privacy. There is support for the authentication protocols HMAC-MD5-96, HMAC-SHA-96, and optional support for the encryption protocol CBC-DES.

The View-based Access Control Model (VACM), defined in RFC 3415, shows how to define views that are subsets of the full MIB tree. Access control on a per-user basis can then be implemented for these views.

Because SNMP has a modular structure, changes to individual modules do not impact the other modules directly. This allows you to easily define SNMPv3 over the existing model. For example, to add a new SNMP message format, it is sufficient to upgrade the message processing model. Furthermore, because it is needed to support SNMPv1 and SNMPv2 messages as well, it can be achieved

[3] ASN.1 BER specifies the Basic Encoding Rules for OSI Abstract Syntax Notation One, according to ISO 8825.

Archive

by adding the new SNMPv3 message module into the message processing subsystem. Figure 17-8 illustrates this structure.

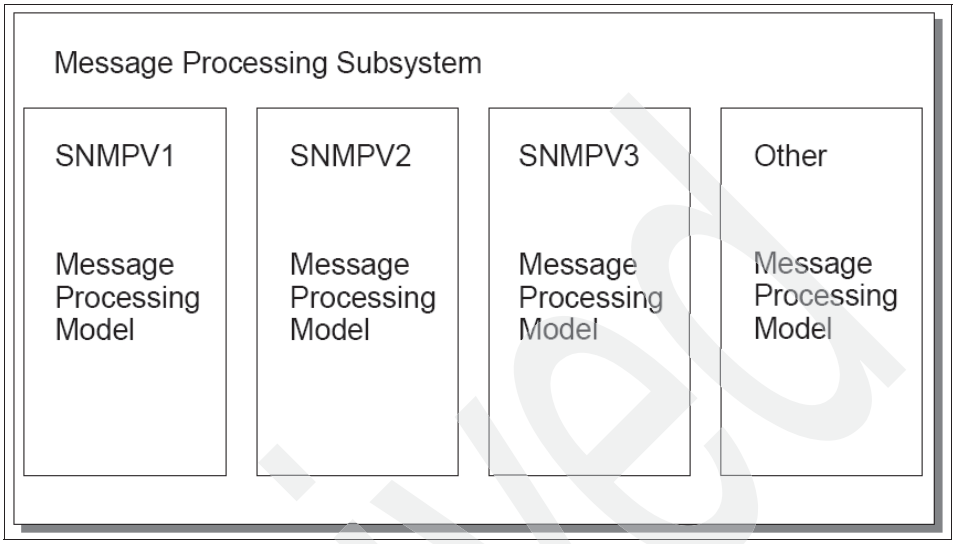


Figure 17-8 The SNMP message processing subsystem

17.1.8 Single authentication and privacy protocol

The authentication protocol provides a mechanism by which SNMPv3 management communications, transmitted by a party, can be reliably identified as having originated from that party.

The privacy protocol provides a mechanism by which SNMPv3 management communications transmitted to a party are protected from disclosure.

Principal threats against which the SNMPv3 security protocol provides protection are:

- Modification of information
- Masquerade
- Message stream modification
- Disclosure

The following security services provide protection against these threats:

Data integrity Provided by the MD5 message digest algorithm. A 128-bit digest is calculated over the designated portion of a SNMPv3 message and included as part of the message sent to the recipient.

Data origin authentication	Provided by prefixing each message with a secret value shared by the originator of that message and its intended recipient before digesting.
Message delay or replay	Provided by including a time stamp value in each message.
Data confidentiality	Provided by the symmetric privacy protocol that encrypts an appropriate portion of the message according to a secret key known only to the originator and recipient of the message. This protocol is used in conjunction with the symmetric encryption algorithm, in the cipher block chaining mode, which is part of the Data Encryption Standard (DES). The designated portion of an SNMPv3 message is encrypted and included as part of the message sent to the recipient.

17.2 The NETSTAT utility

The NETSTAT utility is a command available on most platforms that enables a user to list the sockets in use on a system. The information returned by the command is only for the local host, and there is no provision for monitoring remote hosts using this utility.

The most common uses for NETSTAT are:

- Determining how many sockets are currently open on a system
- Determining what application owns a particular socket
- Diagnosing TCP/IP problems
- Diagnosing routing problems

The NETSTAT command can be issued with or without parameters. Without parameters, the output generated by the command typically lists all of the active UDP and TCP connections in the system's connection table. Options can be added to filter the output, or to request additional information. Because NETSTAT is not RFC defined, the specific options employed by different implementations vary. However, there is a common set of options that remain constant among most NETSTAT implementations.

17.2.1 Common NETSTAT options

Common NETSTAT options include:

-r / **-route**	Displays the routing table currently used by the TCP/IP application.
-i / **-interface**	Displays a list of interfaces, and their states.
-l / **-listening**	Displays only sockets on which an application is listening.
-a / **-all**	Displays all connections (typically, this is the default).
-s / **-statistics**	Displays the statistics for each protocol.
-t / **-timer**	Displays timer information.
-v / **-verbose**	Displays the output in verbose mode.
-f / **-family**	Displays the address family of the connections.

17.2.2 Sample NETSTAT report output

Example 17-1 is a sample of a NETSTAT -all command and illustrates what is usually output by the default implementation of the utility.

Example 17-1 NETSTAT -all command output

```
:\> NETSTAT -a

TCPIP Name: TCPIP                                              13:11:51
User Id  Conn     Local Socket          Foreign Socket         State
-------  ----     ------------          --------------         -----
FTPD1    00064A00
10.44.36.163..21       10.76.141.227..1780    Establsh
FTPD1    00000039 0.0.0.0..21           0.0.0.0..0             Listen
PSF06A   00064B75 10.44.36.163..1384     10.27.172.17..9100     SynSent
SMTP     00000037 0.0.0.0..25           0.0.0.0..0             Listen
SNMPD    00000031 0.0.0.0..1026         0.0.0.0..0             Listen
TCPIP    0006421F
10.44.36.163..23       10.27.204.195..1055    Establsh
SMTP     00000038 0.0.0.0..1028         *..*                   UDP
SNMPD    00000030 0.0.0.0..161          *..*                   UDP
```

The columns of the output, as well as in most implementations, are defined as follows:

User Id	The application or user that is using the socket.
Conn	The connection identification number.

Local Socket The local IP address and port over which the connection is active.

Foreign Socket The remote IP address and port over which the connection is active.

State The state of the connection. Most implementations use some form of the following values for state:

- CloseWait
- Closed
- Established
- FinWait_1
- FinWait_2
- LastAck
- Listen
- SynReceived
- SynSent
- TimeWait
- UDP
  (Because UDP is a connectionless protocol, they cannot be listed in a particular state. As such, NETSTAT simply indicates that they are UDP sockets.)

Additional information about these states can be found in RFC 0793.

Additionally, Example 17-2 is a sample routing table generated by NETSTAT.

Example 17-2 Sample routing table

```
:\> NETSTAT -r

TCPIP Name: TCPIP                                    13:25:04
Destination      Gateway          Flags     Refcnt  Interface
-----------      -------          -----     ------  ---------
Default          10.44.36.129     UGS       001504  INTRF1
Default          10.44.36.129     UGS       000006  INTRF2
10.44.36.128     0.0.0.0          US        000003  INTRF1
10.44.36.128     0.0.0.0          US        000000  INTRF2
10.44.36.129     0.0.0.0          UHS       000000  INTRF1
10.44.36.129     0.0.0.0          UHS       000000  INTRF2
10.44.36.163     0.0.0.0          UH        000000  VIPAL1
10.44.36.164     0.0.0.0          UH        000000  INTRF1
10.44.36.165     0.0.0.0          UH        000000  INTRF2
127.0.0.1        0.0.0.0          UH        000002  LOOPBACK
```

Again, the columns above are defined as follows:

Destination	The route described by the report.
Gateway	The gateway (if any) used to reach this route.
Flags	Attributes of the route. Possible values include: • G: The route uses a gateway. • U: The interface over which the route travels is up. • H: Only a single host can be reached through this route. • D: The route was dynamically created. • M: The route's table entry was modified by an ICMP redirect message. • !: The route is a reject route; datagrams will be dropped.
Refcnt	The number of connections using this route.
Interface	The interface used by the route.

17.3 RFCs relevant to this chapter

The following RFCs provide detailed information about the management protocols and architectures presented throughout this chapter:

- RFC 0793 – Transmission Control Protocol (September 1981)
- RFC 1028 – Simple Gateway Monitoring Protocol (November 1987) Historic
- RFC 1052 – IAB recommendations for the development of Internet network management standards (April 1988)
- RFC 1085 – ISO presentation services on top of TCP/IP based internets (December 1988)
- RFC 1095 – Common Management Information Services and Protocol over TCP/IP (CMOT) (April 1989)
- RFC 1155 – Structure and identification of management information for TCP/IP-based internets (May 1990)
- RFC 1156 – Management Information Base for network management of TCP/IP-based internets (May 1990)
- RFC 1157 – Simple Network Management Protocol (SNMP) (May 1990)
- RFC 1189 – Common Management Information Services and Protocol for the Internet (CMOT and CMIP) (October 1990)

Archived

- RFC 1213 – Management Information Base for Network Management of TCP/IP-based internets: MIB-II (March 1991)
- RFC 1215 – Convention for defining traps for use with the SNMP (March1991)
- RFC 1239 – Reassignment of experimental MIBs to standard MIBs (June 1991)
- RFC 1351 – SNMP Administrative Model (July 1992)
- RFC 1352 – SNMP Security Protocols (July 1992)
- RFC 1441 – Introduction to version 2 of the Internet-standard Network Management Framework (April 1993)
- RFC 1592 – Simple Network Management Protocol Distributed Protocol Interface Version 2.0 (March 1994)
- RFC 1748 – IEEE 802.5 MIB using SMIv2 (December 1994)
- RFC 1901 – Introduction to Community-based SNMPv2 (January 1996)
- RFC 1909 – An Administrative Infrastructure for SNMPv2 (February 1996)
- RFC 1910 – User-based Security Model for SNMPv2 (February 1996)
- RFC 2578 – Structure of Management Information Version 2 (SMIv2) (April 1999)
- RFC 2579 – Textual Conventions for SMIv2 (April 1999)
- RFC 2580 – Conformance Statements for SMIv2 (April 1999)
- RFC 2742 – Definitions of Managed Objects for Extensible SNMP Agents (January 2000)
- RFC 2863 – The Interfaces Group MIB (June 2000)
- RFC 3410 – Introduction and Applicability Statements for Internet-Standard Management Framework (December 2002)
- RFC 3411 – An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks (December 2002)
- RFC 3412 – Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) (December 2002)
- RFC 3413 – Simple Network Management Protocol (SNMP) Applications (December 2002)
- RFC 3414 – User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) (December 2002)
- RFC 3415 – View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) (December 2002)

- RFC 3416 – Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP) (December 2002)
- RFC 3417 – Transport Mappings for the Simple Network Management Protocol (SNMP) (December 2002)
- RFC 3418 – Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) (December 2002)
- RFC 3584 – Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework (August 2003)
- RFC 4022 – Management Information Base for the Transmission Control Protocol (TCP) (March 2005)
- RFC 4113 – Management Information Base for the User Datagram Protocol (UDP) (June 2005)
- RFC 4293 – Management Information Base for the Internet Protocol (IP) (April 2006)

Archived

Archived

18

Wireless Application Protocol

Wireless Application Protocol (WAP) is the *de facto* worldwide standard for providing Internet communications and advanced services on digital mobile devices, such as handheld phones, pagers, and other wireless devices. This protocol is an open, global specification that enables users of these digital devices to securely access and interact with Internet, intranet, and extranet applications and services.

Many phones and PDAs currently have multimedia capabilities that include:

- Retrieving e-mail
- Accessing data from company databases
- Executing stock trading
- Paying bills
- Making travel reservations
- Online requests for maps and driving directions
- Running other exciting applications

By 2010, there will be more than 2.3 billion individual wireless subscribers worldwide, according to "World Mobile Subscriber Markets 2005,"[1] with 1.1 billion of those subscribers using third generation (3G) services. As technologies

© Copyright IBM Corp. 1989-2006. All rights reserved.

advance, we now have fourth generation (4G) services. These boast new promises for mobile internetwork services, and currently are successfully implemented in several places around the world.

Based on these forecasts, wireless service providers, manufacturers of handheld devices, and software development companies see a huge new market of information-hungry users.

In this chapter, we discuss the WAP version 2 standard. WAP version 2 brings together open Internet standards and mobile networking as it adds optimized support for TCP and HTTP, Extended Hypertext Markup Language (XHTML), and Transport Layer Security.

The WAP Forum was founded in June 1997, by Ericsson, Motorola, Nokia, and Phone.com. It has drafted a global wireless specification for all wireless networks. The forum contributed to various industry groups, such as wireless service providers, handset manufacturers, infrastructure providers, and software developers. The forum has also provided the specifications to standards bodies such as ARIB, CDG, ECMA, ETSI, TIA, and W3C.

The WAP Forum consolidated into the Open Mobile Alliance (OMA) and does not exist as an independent organization.

Most of the content of this chapter has been extracted from the OMA specifications.

For more information about WAP specification work, refer to:

`http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html`

At this Web site, you can find the latest work and all of the specification documents.

[1] Visant Strategies, Inc. (September 2005)

18.1 The WAP environment

Wireless data networks present many communications challenges compared to wired networks. For example, the wireless communication environment is constrained because wireless links have:

- Less bandwidth
- More latency
- Less connection stability
- Less predictable availability

As a result, protocols that provide wireless applicability must be tolerant of these types of problems. In addition, machines using these wireless links are limited in terms of processing power and other functionality. For example, wireless devices lack the traditional desktop GUI like the Web browser. Other fundamental limitations for handheld, wireless devices include:

- Less powerful CPUs
- Less memory
- Restricted power consumption
- Smaller displays
- Different input devices (for example, phone keypad and voice input)

The first WAP specification, named WAP version 1, solved basic user application requirements, which were primarily related with information inquiries rather than transaction processing. For example, users could only view what e-mails were received instead of reading the contents of the e-mail on the small screen, or looking at stock quotes rather than doing stock trading. However, with the new WAP version 2, it is assumed that other tasks, such as executing banking transactions or electronically signing documents, are demands that have to be considered. This requires reliable data transmission, message privacy, integrity, and authentication. As a result of the effort of many on the WAP Forum and OMA, these functions are included as part of the specifications of WAP version 2.

18.2 Key elements of the WAP specifications

WAP defines an open standard architecture and a set of protocols to implement wireless Internet access. The key elements of the WAP specification include:

- Definition of the WAP programming model (see Figure 18-1 on page 658).

- The Extended Hypertext Markup Language Mobile Profile (XHTMLMP).
- The Wireless Markup Language (WML).
- A lightweight protocol stack (see “Components of the WAP architecture” on page 661).
- A framework for wireless telephony.

18.3 WAP architecture

The WAP architecture illustrated in Figure 18-1 describes:

- The WAP client (the handheld device or WAP terminal)
- The application server

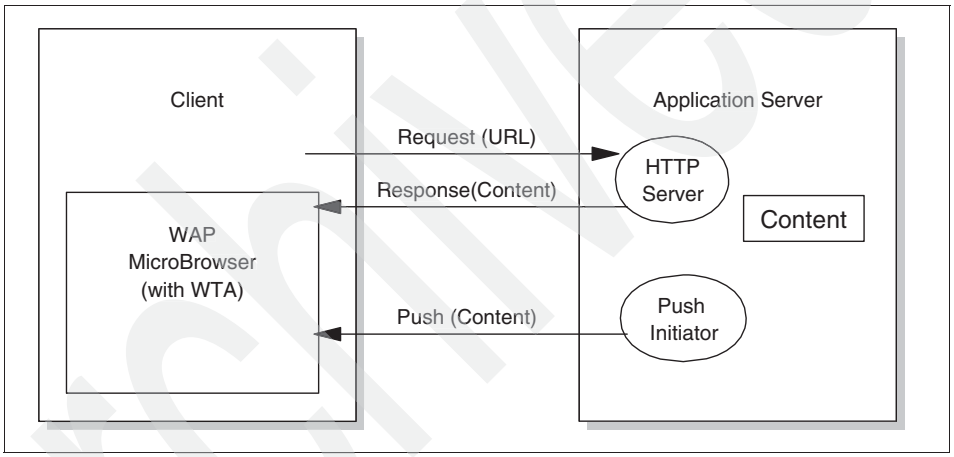


Figure 18-1 WAP programming model

This programming model is the Web programming model with some extensions and enhancements to match the characteristics of the wireless clients. This programming model adds two enhancements to the Web programming model:

- Push
- Wireless Telephony Support (WTA)

Programmers will see it as an advantage because they will meet a well-known interface based on client/server technology.

You can find more about WAP architecture in OMA WAP-210-WAPArch-20010712-a specification.

Archived

Feature and performance-enhancing proxies

The WAP model recommends the use of proxies to optimize the connection between the wireless clients and the Web (see Figure 18-2).

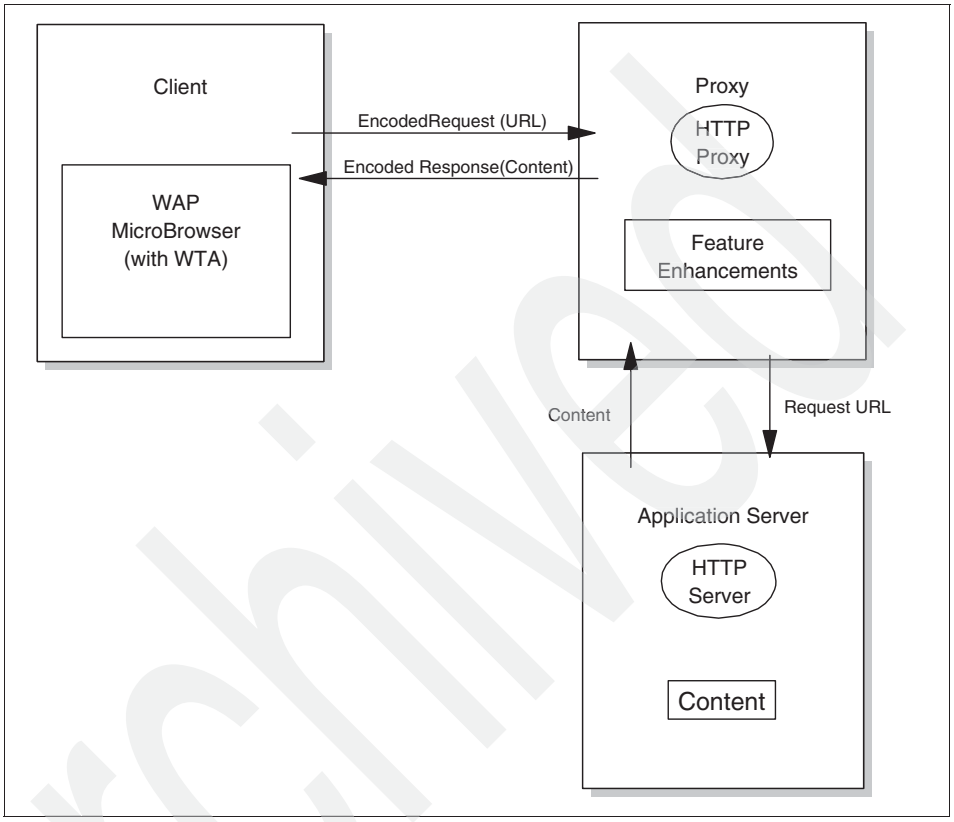


Figure 18-2 Feature and performance-enhancing proxy

A variety of functions are then provided by the WAP proxy, including:

Protocol gateway Translates requests from a wireless protocol stack (for example, the WAP 1.x stack—WSP, WTP, WTLS, and WDP) to the WWW protocols (HTTP and TCP/IP). Also can perform DNS lookups for the URLs requested by the client.

Content encoders and decoders
Translates WAP content for better utilization of the underlying link, reducing the bandwidth use by different compression techniques.

Archived

User Agent Profile Manager
: Used mainly to communicate the client's device capabilities and personal preferences to the server applications.

Caching proxy
: Caching frequently accessed resources, a caching proxy can improve network utilization.

The use of proxy technology facilitates the access to many different Internet applications and services and Web developers are able to facilitate content compatible with a large number of mobile devices.

WAP network elements

WAP clients can communicate with application servers directly or through proxies. WAP clients support the proxy selection mechanism that allows them to use the most appropriate proxy for a given service or to connect directly to that service as necessary. If proxies are used, they translate between WAP and Web protocols (HTTP, TCP), thereby allowing the WAP client to submit requests to the origin server.

To optimize the communication between device and application server, proxies can be located in order to provide feature enhancements coupled to the wireless network (for example, telephony, location and provisioning) or proxies can be located in a secure network to provide a secure channel between wireless device and the secure network.

The supporting servers provide support for additional functions such as User Agent Profile (UAProf, see 18.8, "User Agent Profile (UAProf)" on page 671), provisioning, PKI, and so on.

Archive

Components of the WAP architecture

The layered design for the protocol stack on the WAP architecture allows an extensible and scalable application development environment for mobile devices. Each layer provides a set of functions or services, or both, to the other layers above and to the other services and applications. Figure 18-3 shows the WAP stack architecture.

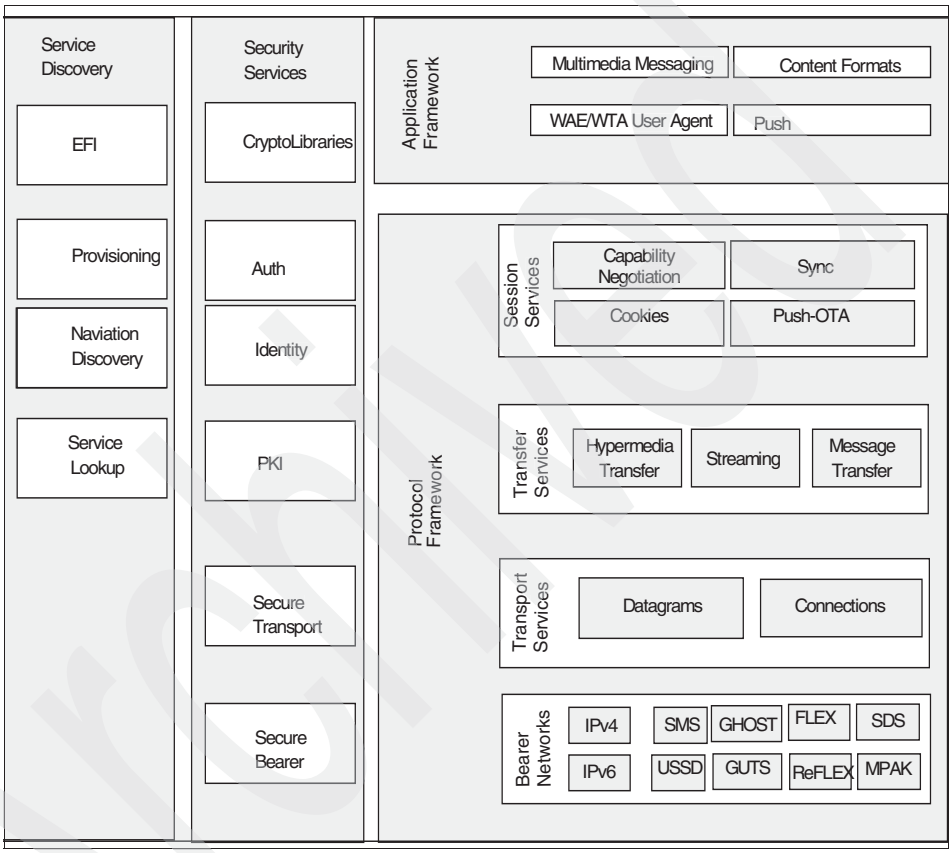


Figure 18-3 WAP stack architecture

By separating the service interfaces from the protocols that provide those services, the WAP architecture allows the evolution of the specifications and the selection of the most appropriate protocol for any given context. Services in the the stack can be provided by more than one protocol.

Archive

Here, we give a brief description of the protocol stack:

Bearer networks Different bearer networks are allowed. As the transport layer acts as an interface between the bearer service and the upper layers of the WAP stack, the transport layer specifications defines the supported bearers and the functions implemented to access each bearer.

Transport services This service is in charge of transportation of unstructured data across the underlying bearer networks, creating an consistent abstract implementation across all of the supported bearers. The transport services include datagram and connection services among others.

Transfer services This service provides the implementation for the structured data transfer of information between network elements. Most popular services included are hypermedia transfer (such as HTTP, Wireless Session Protocol, and Wireless Transaction Protocol), streaming data (such as audio and video), and message transfer (such as Multimedia Message Services Encapsulation).

Session services This service establishes a shared state between network elements. For example, services include capability negotiation, push over the air, sync service, and cookies.

Application framework This provides the general purpose application environment based on a combination of World Wide Web (WWW), Internet, and mobile technologies. The primary objective of the application framework is to establish an interoperable environment that will allow service providers to build applications and services that will be supported by a wide variety of different platforms in an efficient manner. The application framework includes, among others, push, multimedia messaging, and content formats.

Security services This is a fundamental part of the WAP architecture and it is present in many of its layers. The services commonly offered at this layer are authentication, privacy, integrity, and non-repudiation.

Service discovery This is a fundamental part of the WAP architecture and this can also be found at many layers. Some examples of this are the external functionality interface (EFI), provisioning, navigation discovery, and service lookup.

Other services and applications

The WAP architecture allows another set of services or applications to be implemented through a set of well-defined interfaces. For example, applications such as e-mail, calendars, electronic commerce, or services such as yellow pages or maps and directions are being developed to use the WAP protocols.

18.4 Client identifiers

The client identifier (client ID) identifies a mobile device in the network. The client ID can be assigned by the proxy server component or it can be formed by using the syntax defined in the client ID specification.

The Client ID specification is defined in the OMA WAP-196-ClientID-20010409-a specification document. This specification does not exclude any other identification alternatives.

The syntactical format of client ID is defined using Augmented Backus-Naur Form.

18.5 Multimedia messaging system (MMS)

The multimedia messaging system (MMS) defines application-level protocol activities that take place to realize the MMS service within the WAP environment to provide messaging operations, allowing different kinds of content and media types to be shared in the network.

The MMS is intended to provide support for the user need to share new media capabilities present in mobile devices, such as taking pictures or recording video.

The multimedia messaging system is viewed as an asynchronous delivery system. This is the same as in e-mail delivery systems and some other Internet applications. This type of delivery system allows an optimum use of the network capabilities, both for the client and the network servers.

A special characteristic of MMS is the ability to support messaging activities with other available messaging systems in the Internet.

For more information about MMS, refer to OMA WAP-205-MMSArchOverview-20010425 specification.

18.6 WAP push architecture

The design of the client/server model allows only the clients to make requests of a service from the server. The server responds by transmitting the information to the client. This type of information exchange is called pull technology, because the client pulls the information from the server.

The WWW method is a typical example of the pull technology. The client enters an URL to retrieve information from the Web server. The Web server sends the information to the client for displaying the Web page on the Web browser.

Following traditional Web technology, wireless service providers could not send messages to the client without a request being made. In order to make this service possible, the push architecture was designed.

In contrast, the WAP architecture defines, in addition, a push technology. Both are shown Figure 18-4. This technology is also based on the client/server model, but there is no explicit request from the client to the server. The push transactions are initiated by the server only. It is used to transmit information to a WAP terminal without a previous user action.

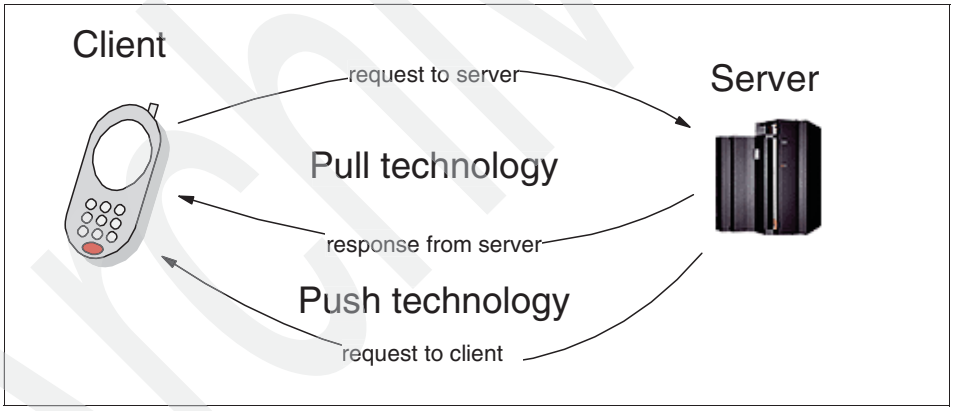


Figure 18-4 Push/pull technology

18.6.1 Push framework

Within TCP/IP, information requests to a server can be initiated from the client only. However, information service providers also need a way to contact clients without an outstanding client request. For example, a mail service provider wants to notify a WAP client that e-mails are in the client's mailbox, and also wants to provide all the senders' names, or to deliver the newest stock quotes, headlines news, and weather and traffic conditions to which the WAP client has subscribed.

This kind of transaction, where the server is the initiator of the connection with a WAP client, is a new feature. It is designed under the name push initiator, as illustrated in Figure 18-5.

Because the WAP client is located in the WAP domain and the server is in the Internet, both do not share the push initiator protocol. Therefore, again, gateway support is needed to provide this functionality. A gateway that performs pushing is called a push proxy gateway (PPG).

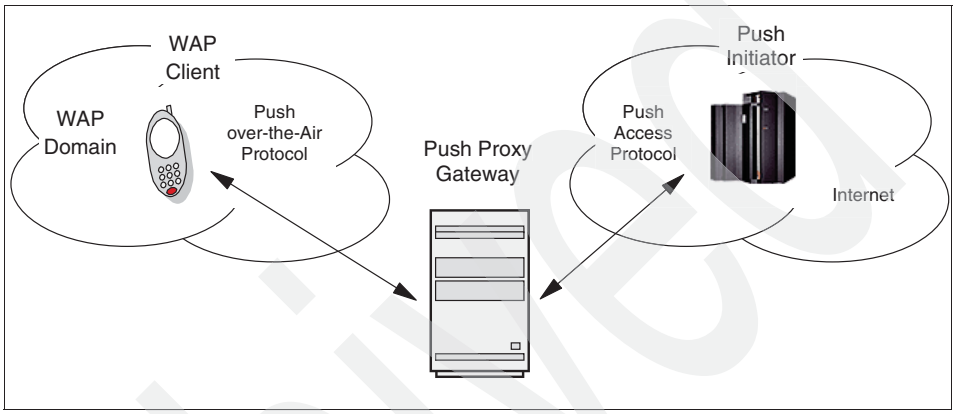


Figure 18-5 Push technology

If the push initiator wants to send out information to WAP clients, it contacts the push proxy gateway (PPG). It uses traditional Internet protocols together with the push access protocol (PAP). The PPG forwards the pushed content to the WAP domain, where it is transmitted over the air using the push over-the-air protocol (OTA).

The PPG might have the capability to tell the push initiator about the state of the delivery of the message to the client. But this can take some time, because the WAP client might not be online. If the client is online, it can accept or reject the pushed content.

The PAP uses XML messages, which can be tunneled through HTTP through the Internet.

18.6.2 Push proxy gateway (PPG)

The PPG is the access point for content pushes from the Internet to the mobile network. It does everything that is necessary for this kind of operation, such as:

- Authentication
- Security

- Client control

As owner of the gateway, it decides who gets access to the WAP domain and who is allowed to push messages.

Overview of PPG services

The services of the push proxy gateway include:

- Push initiator identification authentication, access control
- Parsing of and error detection in content control information
- Client discovery services
- Address resolution
- Binary encoding and compilation of certain content types to improve OTA
- Protocol conversion

Access from the Internet side

PPG accepts content from the Internet side, which is divided into several parts using multipart/related content type. The first part contains the information for the PPG, for example:

- Recipient information
- Timeouts
- Callback requests

The PPG acknowledges the parsing of this control information. If the push initiator requests, it can also call back a pushing server when the final status of the push submission to the WAP client has been reached, and will report about the status (delivered, cancelled, or expired).

The PPG pushed content delivery

PPG tries to find the correct WAP client and, if successful, delivers the content using the push over-the-air protocol. The trial to deliver is limited by the timeout specified for the client. This timeout can be set by the push initiator or the policies of the mobile operator.

PPG can implement addressing aliasing schemes to enable special multicast and broadcast functions. This requires the translation of special addresses into a broadcast operation. The push initiator can also query client capabilities and preferences to create a better formatted content for a particular WAP terminal.

18.6.3 Push access control protocol (PAP)

The push initiator uses PAP to push content through a PPG to the WAP domain. PAP is carried over an HTTP 1.1 tunnel. PAP carries an XML-style entity.

PAP operations

PAP supports the following operations:

- Push submission (initiator to PPG)

  The push message contains a control entity, which provides delivery instructions for the PPG, a content entity, which is the textual content for the WAP terminal, and, optionally, a capability entity.

  Depending on the message type (WML or WMLScript), PPP converts this message into a more bandwidth-optimized form before forwarding over-the-air (OTA). Also, encryption can be used when sending the message to the client.

- Result notification (PPG to initiator)

  If the push initiator has requested confirmation of successful delivery, a notification message is returned to the push initiator. This gives the push initiator the awareness that the WAP client has also acknowledged the successful delivery to the PPG.

- Push cancellation (initiator to PPG)

  This XML-entity is sent from the push initiator to the PPG, requesting that a previously submitted content be cancelled. The PPG responds if the cancellation was successful or not.

- Status query (initiator to PPG)

  The push initiator requests the status of delivery during the 2-way process to the WAP client:

  - First way: Push initiator → PPG
  - Second way: PPG → client

- Client capabilities query (initiator to PPG)

  The push initiator requests, from the PPG, information about the capabilities of a particular WAP terminal. The PPG responds with a multipart/related message in two parts:

  - The first part is about the execution of the message itself.
  - In the second part, the capabilities of the WAP terminal defined by the User Agent Profiles group is reported.

Archived

18.6.4 Service indication

The service indication (SI) content type enables the sending of notifications to users in an asynchronous manner (for example, new e-mails). An SI contains a short message and a Uniform Resource Identifier (URI), indicating a service.

The message is presented to the user on reception. The user can now select to either start the service immediately or postpone it for a later execution. If the SI is postponed, the client stores it in order to enable the user for later SI handling.

18.6.5 Push over-the-air protocol (OTA)

OTA is responsible for transmitting the content from the PPG to the client's user agent over the wireless network.

OTA can use a WSP session to deliver the content. However, a WSP session works in a connection-oriented mode and has to be established by a client prior to delivery of the pushed content. In this case, where there is no active WSP session, a session initiation application in the client has to establish the session. This new function in the client works like a server that listens to session requests from OTA servers and responds by setting up a WSP session for the push purpose.

The client can verify the identity information of the OTA server against a list of such servers before attempting any push session.

18.6.6 Client-side infrastructure

A connection-oriented push requires an active WSP session. As explained in 18.6.5, "Push over-the-air protocol (OTA)" on page 668, the client needs a special session initiation application (SIA) in order to set up the push session with the PPG. After receiving a session request from PPG, SIA establishes a session with the PPG and reports which client applications accept content over the newly opened session. SIA can also ignore the session request from the PPG if there is no suitable application available or installed.

When the client receives pushed content, a dispatcher looks at the push message header to determine its destination application. This dispatcher is responsible for rejecting content if the destination application is not installed. It is also responsible for confirming operations to the PPG.

18.6.7 Security

In a trusted environment, several questions can arise, for example:

- How can the push initiator be authenticated?
- What role does PPG play in a security and trusted model?
- What are the access control policies for a push initiator and pushed content?
- How can a client authenticate something if it has no certificates?

Authenticating a push initiator

Some of the following solutions can be used to implement a security environment between a push initiator and the PPG:

- Use of session level certificates (TSL and SSL)

  If the push content traverses the Internet between the push initiator and PPG, TSL or SSL can be used.

- Use of object-level certificates

  Certificates can be used to sign and encrypt the pushed content on an end-to-end basis. This strengthens the level of confidence in the content authenticity at the client's end.

- HTTP authentication

  The basic authentication through user ID and password is available. In addition, HTTP authentication, for example, based on digests, can be implemented.

- Combination of technologies

  Another approach is combining technologies by using a TLS/SSL session with a PPG, while HTTP authentication can be used to authenticate the push initiator. Signed and encrypted content can then be sent over this authenticated session.

Client authentication

If a client and a PPG are able to create a trusted environment, the PPG can authenticate a push initiator on behalf of that client, that is, trust can be transitive. A trust situation can be established between client and PPG by maintaining a list of trusted PPGs in a client system. Push initiators own a certificate or public key for end-to-end authentication of the origin server.

Archived

18.7 The Wireless Application Environment (WAE2)

The Wireless Application Environment (WAE2), defined by the OMA wap-236-waespec-20020207-a specification, provides an environment on which programmers and developers can construct applications for use on wireless devices. Specifically, WAE2 is intended to provide optimized access to resources and applications, while still remaining within the limits of constricted CPU, battery life, and bandwidth. It is backward-compatible with the original WAE (from WAP version 1). Figure 18-6 illustrates where WAE2 fits within the WAP architecture.

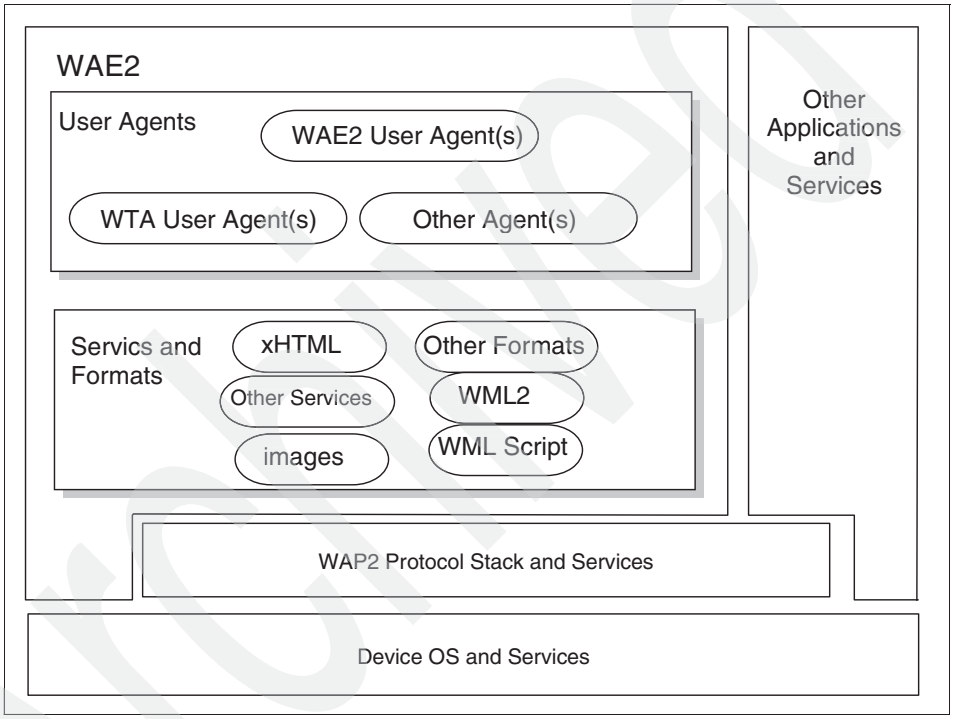


Figure 18-6 WAE2 client components

WAE2 defines a set of content formats for interoperable data exchange. The way data is exchanged depends on the data and the targeted WAE2 user agents. The two main formats are the encoded WML and the WMLScript byte code formats. These formats are mainly designed for WAP terminals to minimize computational

power and for lower bandwidth usage. There are, in addition, formats for data types such as:

- Images

  It supports multiple choices of pixel depth, color space tables, small encoding, very low CPU and RAM encoding and presentation demands, availability of common tools, and so on.

- Multipart messages

  It supports a multipart-encoding scheme optimized for exchanging multiple typed content over WSP (see 18.9.5, “Wireless Session Protocol (WSP)” on page 682).

- User agent-specific formats

  This is a function that supports two additional formats for exchanging data among user agents: electronic business card (vCard 2.1), and electronic calendaring and scheduling exchange format (vCalender 1.0) specified by IMC.

  WAE also defines WTA-specific formats to support wireless telephone applications.

18.8 User Agent Profile (UAProf)

One of the aspects inherent in a wireless network is the existence of heterogeneous mobile hardware, comprising a variety of different capabilities. Therefore, it becomes possible that origin servers deliver content to mobile clients that are beyond the client’s capabilities to display. To prevent such an occurrence, the User Agent Profile (UAProf) was created and defined in the OMA WAP-248-UAProf-20011020-a specification.

The UAProf allows a client to send data regarding the client’s capabilities to an origin server, thereby indicating what type of content the server can and cannot send. Information contained in the UAProf includes:

- Hardware-specific data, such as the device manufacturers, hardware versions, image capability, and so on
- Software-specific data, such as the operating system, types of encoders, applications present, and so on
- Browser-specific data, such as the mark-up and scripting languages that are supported
- Network-specific data, such statistics on the device's performance capabilities

- WAP-specific data, such as the WML capabilities and WAP version

Additionally, an origin server or proxy server can cache this information to expedite the a client's requests.

18.9 Wireless protocols

The wireless application protocol consists of numerous subprotocols, each providing a different functionality for wireless applications. This section describes these protocols and the scenarios for which they are appropriate.

18.9.1 Wireless Datagram Protocol (WDP)

WDP provides a consistent service to the upper layers (security, transaction, and session) of the WAP architecture. It is defined in the OMA WAP-259-WDP-20010614-a specification, and allows applications to operate transparently over different available bearer services. It communicates transparently over the different bearer services supported by multiple network types. WDP is a connection-less, unreliable datagram service. It supports port number addressing. The port number points to the higher-layer level of WDP. This can be WTP, WTLS, WSP, or an application.

In order to support the different bearer services with its specific capabilities and characteristics, an adaptation is required to keep WDP as a common layer for the various bearer services. Therefore, WDP, with its type of adaptation layer, cooperates with its underlying bearer layer. Figure 18-7 on page 673 shows the general WDP structure.

Archived

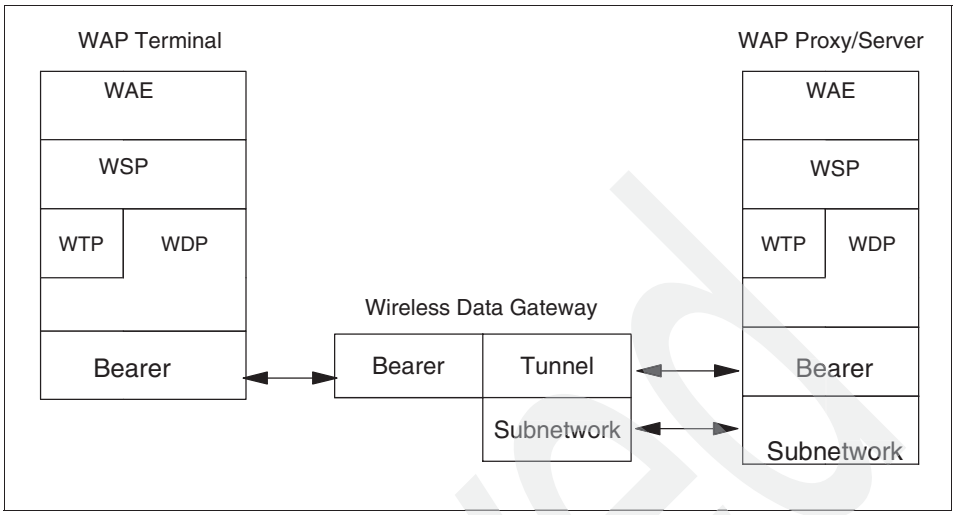


Figure 18-7 General WDP architecture

WDP messages are sent by the WAP terminal to the wireless data gateway using the bearer services. The wireless data gateway has the choice to pass WDP packets on to the WAP proxy/server through a tunneling protocol, which is the interface between the gateway that provides bearer service and the WAP proxy server. For example, if the bearer service was an GSM SMS, the gateway would be a GSM SMSC and would support a specific tunneling protocol to interface the SMSC to other servers. It is also possible to use a subnetwork as a common technology in order to connect two communication devices. This connection can be, for example, through a wide area network based on TCP/IP or frame relay, or a LAN operating TCP/IP over Ethernet. The WAP proxy/server might offer application content or might operate as gateway between the wireless WTP protocols and the wired Internet.

When used over an IP network layer, UDP is used instead of WDP.

WDP service primitives

Service primitives are used to control the transaction traffic between the layers of the client and server. These service primitives have a similar syntax to that described for WSP:

```
X - generic name. type (parameters)
```

Where X designates the layer providing the service. For the WDP layer, it is T.

The primitive types and their abbreviations are listed in Table 18-1.

Table 18-1 Service primitives

Type	Abbreviation	Description
Request	req	Used when a higher level requests a service from a lower level.
Indication	ind	Used to notify the next higher layer of activities to the peer (such as an invocation request) or to the provider of the service (such as a protocol generated event).
Response	res	A layer uses the response to acknowledge receipt of indication from the next lower layer.
Confirm	cnf	The layer providing the requested service uses confirm to report successfully completion of the activity.

There are two primitives for the service layer:

T-DUnitdata Transmits data as datagram.

T-DError WDP can also receive a T-DError primitive if the requested transmission cannot be executed by the WDP protocol layer.

Mapping WDP for IP

User Datagram Protocol (UDP), as a connection-less datagram service, is used as the WDP protocol for any wireless bearer network where IP is used as a routing protocol. UDP provides port-based addressing (destination and source port). IP provides the segmentation and reassembly.

18.9.2 Wireless Profiled Transmission Control Protocol (WP-TCP)

One of the specifications newly introduced by the transport layer of the WAP2 architecture is a Wireless Profiled TCP (WP-TCP) used to provide connection-oriented services. The implementation of WP-TCP is defined in the OMA WAP-225-TCP-20010331-a specification. Benefits realized from this new implementation include:

- Large data transfer
- End-to-end security
- Adherence to IETF protocols

Use of WP-TCP can use one of two approaches.

The split-TCP approach

The split-TCP approach very closely mirrors the WAP 1.x model. In this approach, a proxy exists between the WAP client and the origin server, and two connections are established. The first connection is established using WP-TCP, and exists between the WAP client and the WAP proxy. The second connection, established over TCP, exists between the WAP proxy and the origin server. This is illustrated in Figure 18-8.

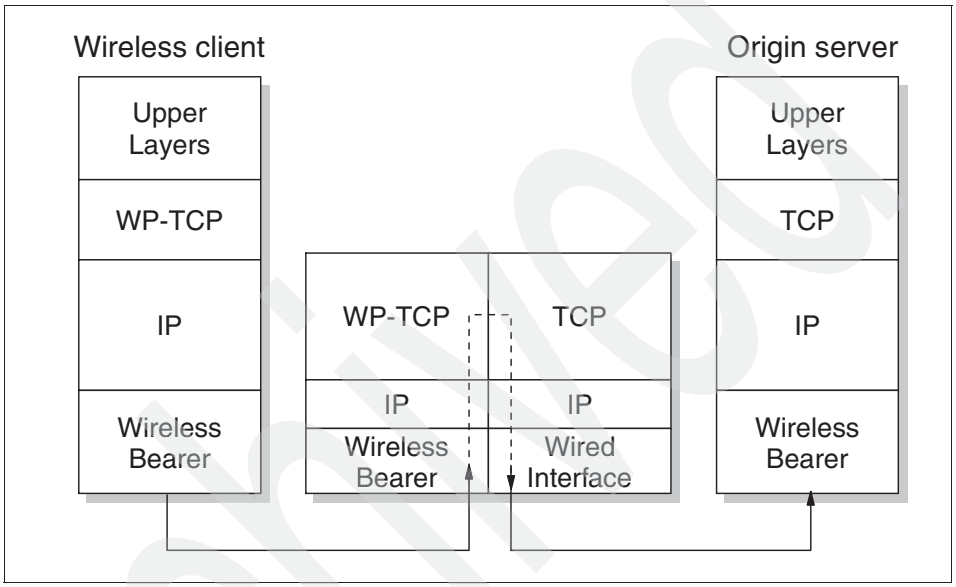


Figure 18-8 The split-TCP approach

Using such an approach allows the WAP client and WAP proxy server to take advantage of optimizations that exist in WP-TCP but not in TCP alone. This is particularly important because there are many aspects on a wireless connection that create an environment not suitable for TCP (these are defined in “WP-TCP optimizations” on page 676).

However, the WP-TCP optimizations are specifically designed to allow the protocol to be more efficient in such an environment. By splitting the connection, WP-TCP is able to operate in the wireless environment, and TCP is able to operate in the wired environment. More information about the specific optimizations is in “WP-TCP optimizations” on page 676.

The end-to-end approach

WP-TCP is compliant with RFCs 0793 and 1122. Therefore, any WP-TCP implementation can also communicate with a TCP communication, though the resulting connection will lack the optimizations designed in WP-TCP. In

implementing an end-to-end connection, WP-TCP behaves like any other TCP application, establishing a connection directly with the origin server. This is illustrated in Figure 18-9.

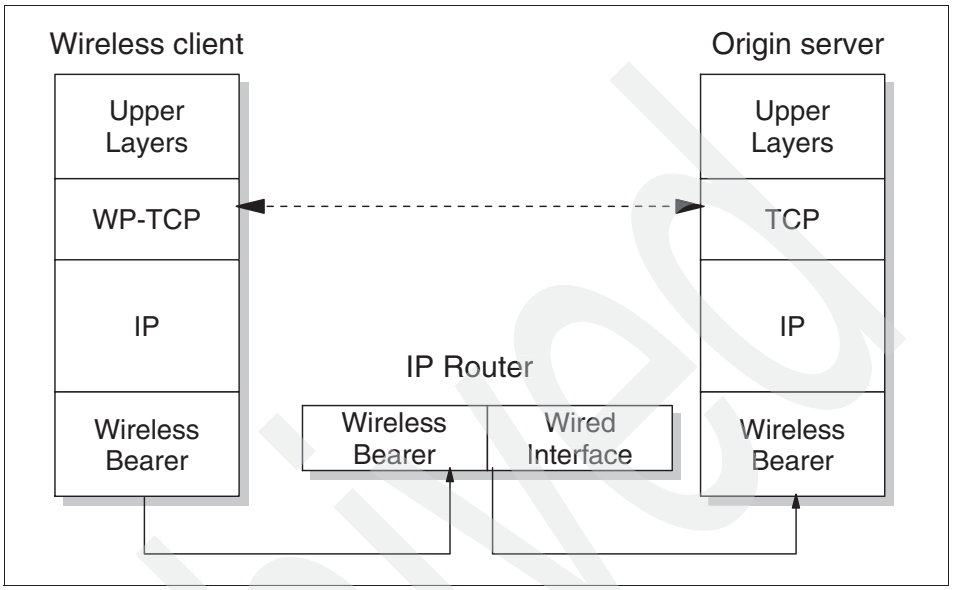


Figure 18-9 The end-to-end approach

The decision to implement the end-to-end approach over the split-TCP approach is usually determined by service, server, and hardware provisioning or the specific needs of the application layer.

WP-TCP optimizations

As noted previously, certain scenarios exist on a wireless network that make a purely TCP connection less desirable. Such scenarios include:

- Frequent packet loss
- Smaller window sizes
- Long periods of silence or connection loss due to exponential back-off retransmission mechanisms
- Redundant transmission
- Periods of disconnection due to lack of coverage

Archived

The profiling of TCP for WP-TCP sought to optimize the protocol to compensate for these scenarios. Specifically, eight optimizations were included in WP-TCP. These are:

- Large window size: Built in to TCP is the use of the Bandwidth Delay Product (BDP), calculated from the available bandwidth and the round-trip time, to calculate the minimum window size. It also calculates the maximum window size using the smallest value of the send or receive buffers. WP-TCP can opt to use the maximum window size, relying on the congestion control mechanisms built into TCP to regulate the data as needed. The WP-TCP specification indicates that implementation of this optimization is recommended, but not required.
- The window scale option: Entities that opt to use the large window size are required to also implement the window scale option defined by RFC 1323. If an entity does not support the large window size optimization, implementation of the window scale optimization is optional.
- Round-trip time measurement: Also defined in RFC 1323, round-trip time measurement (RTTM) is recommended for use by any entity that implements window sizes larger than 64 K. This also requires the use of the time stamp option.
- Large initial window: The standard TCP implementation includes an initial window up to two segments in size. However, WP-TCP employs an extension defined in RFC 2414 that allows WP-TCP to send an initial window of three or four segments. This is limited to 4380 bytes. Implementation of this optimization is optional.
- Path MTU discovery: WP-TCP now supports the procedures, defined in RFCs 1191 and 1981, used to discover the maximum transmission unit (MTU) of any given path. This allows WP-TCP to optimize the amount of data sent in each packet to avoid fragmentation or dropped packets. Implementation of this optimization is recommended, but not required.
- MTU larger than the default IP MTU: If an entity cannot support path MTU discover (for example, if some of the routers in the path do not support the needed ICMP messages), a WP-TCP implementation can assume a value exceeding the default IPv4 or IPv6 values. However, the value assumed must not exceed 1500 bytes, and must reflect the maximum segment size negotiated when establishing the TCP connection.
- Selective acknowledgement: The fast recover algorithm, defined in RFC 2581, tends to be less efficient when attempting to recover from multiple losses in a single window. This scenario is very likely on wireless connections, and therefore, an alternative to the fast recover algorithm was implemented. Selective acknowledgement (SACK) was defined in RFC 2018

and allows WPT-TCP to send acknowledgements selectively, indicating what data needs to be retransmitted. Implementation of this optimization is required.

- Explicit congestion notification (ECN): Defined in RFC 2481, ECN allows a WP-TCP receiver to notify the sender of the data of congestion in the network. This allows the sender to reduce it's congestion window. Implementation of this optimization is optional.

18.9.3 Wireless Control Message Protocol (WCMP)

The Wireless Control Message Protocol (WCMP) very closes mirrors the functionality of the Internet Control Message Protocol (ICMP) defined in RFCs 792 and 2463. Defined in the OMA WAP-202-WCMP-20010624-a specification, it is used by the WDP to provide ICMP-like capability when using wireless bearers that do not support IP. Specifically, WCMP is used to report problems occurring when processing datagrams. Errors reported by WCMP messages can be broken into six types, some of which can be further divided into codes. These are listed in Table 18-2.

Table 18-2 WCMP types and codes

WCMP message	WCMP type	WCMP code
Destination unreachable	**51**	
Communication administratively prohibited		1
Address unreachable		3
Port unreachable		4
Parameter problem	**54**	
Erroneous header field		0
Message too big	**60**	0
Reassembly failure	**61**	
Reassembly time exceeded		1
Buffer overflow		2
Echo request	**178**	0
Echo reply	**179**	0

Archived

WCMP message structure

The message structure of WCMP messages varies based on the type of wireless bearer used by the originator or the message, as well as the type of the WCAMP message. However, every WCMP does adhere to a general format, as defined in Figure 18-10.

Bit/Octet	0	1	2	3	4	5	6	7
1	Type of control message							
2	Code of control message							
3-n	Data fields for WCMP (0...n octets)							

Figure 18-10 General WCMP message structure

18.9.4 Wireless Transaction Protocol (WTP)

The Wireless Transaction Protocol, defined in the OMA WAP-224-WTP-20010710-a specification, provides a mechanism especially designed for WAP terminals with limited resources over networks and with low to medium bandwidth. This technology allows more subscribers on the same network due to reduced bandwidth utilization.

WTP provides unreliable and reliable data transfer based on the request/reply paradigm. Unlike TCP, there is no connection setup and tear down. Compared to TCP, where a SYN and ACK flow is started before the first data is transmitted, WTP carries data in the first packet of the protocol exchange. Additionally, WTP employs a message-oriented model, as opposed to TCP's stream-oriented implementation.

WTP classes of operation

There are three classes of operation, numbering zero through two.

Class 0

Class 0 is defined by the OMA specification as "unconfirmed invoke message with no result message." This is a datagram that can be sent within the context of an existing WSP session (see 18.9.5, "Wireless Session Protocol (WSP)" on page 682). However, it is not intended to be a primary method of sending

datagrams. For that functionality, use WDP instead. The sequence of events in a Class 0 transaction is as follows:

1. An initiator sends an invoke message to the responder. After the message has been sent, the initiator's role in the transaction has ended. No retransmissions are sent of the message.
2. The responder receives the invoke message. No response is sent acknowledging receipt of the invoke message. After the message has been received, the transaction is complete.

Because there are no retransmissions or replies, this type of transaction is considered stateless and cannot be aborted. An additional illustration of a Class 0 transaction can be found in Figure 18-12 on page 685).

Class 1

Class 1 is defined as "confirmed invoke message with no result message." This is used for reliable data push (see 18.6, "WAP push architecture" on page 664), where no response from the destination is expected, though the transaction is still termed a reliable datagram service. The sequence of events in a Class 1 transaction is as follows:

1. An initiator sends an invoke message to the responder.
2. The responder receives the invoke message and returns an acknowledgement (but not a result message). If the initiator does not receive an acknowledgement, the invoke message is retransmitted.

The transaction remains in an active state until the acknowledgement arrives at the initiator. Upon receipt of the acknowledgement, the Initiator considers the transaction to have ended. The Class 1 transaction can be aborted at any point during the sequence of events.

Class 2

Class 2 is defined as "confirmed invoke message with one confirmed result message." In this class, one single request produces a single reply. This comprises the typical request/response interaction model most widely employed by applications. The sequence of events in a Class 2 transaction is as follows:

1. An initiator sends an invoke message to the responder.
2. The responder might send back an acknowledgement to this message. This happens if the responder is not able to send back a result message immediately.
3. The responder sends back a result message. If no acknowledgment had been previously sent to the initiator, this result message implicitly acknowledges the receipt of the invoke message. If the initiator receives no acknowledgement or result message, it retransmits the invoke message.

4. The initiator sends back an acknowledgement upon receiving the result message.

The transaction is considered complete when the responder receives the acknowledgement. If no acknowledgement is received, the responder retransmits the result message. The transaction can be aborted at any time during the sequence of events. An additional illustration of a Class 2 transaction is in Figure 18-15 on page 692.

Layer-to-layer communication

Service primitives are used to control the transaction traffic between the layers of the client and server. These service primitives have a similar syntax as described for WSP:

```
X - generic name. type (parameters)
```

Where X designates the layer providing the service. For the transaction layer, it is TR. The primitive types and their abbreviations are the same as described for WDP (see Table 18-1 on page 674).

There are three primitives for the service to the upper layer:

TR-Invoke Initiates a new transaction.

TR-Result Sends back a result of a previously initiated transaction.

TR-Abort Aborts an existing transaction.

Archived

Example of a WSP-WTP sequence flow

Figure 18-11 depicts the flow of a primitive sequence and shows the relationship between WSP and WTP requests and responses. The flow is based on a Class 2 transaction which, as defined earlier, is a reliable, confirmed message exchange.

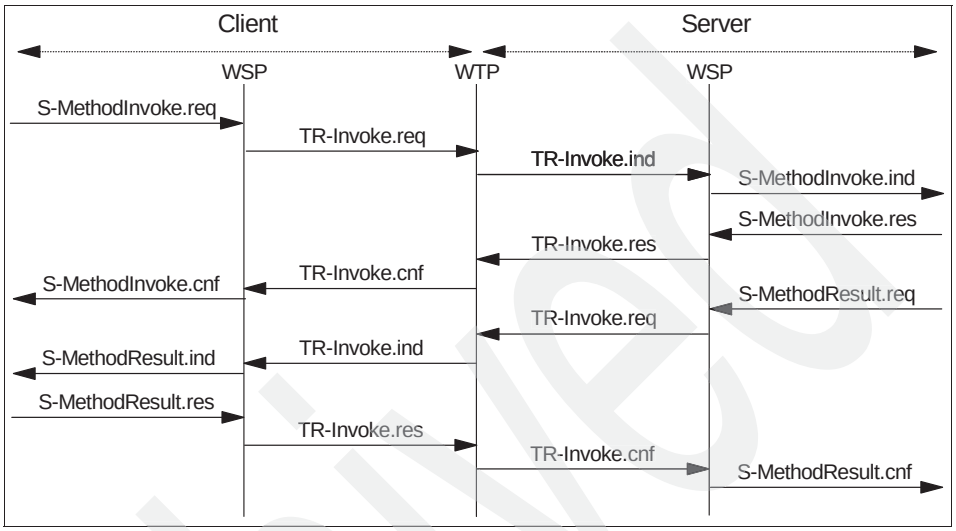


Figure 18-11 WSP-WTP primitive sequence for request-response

The following steps describe the flow of a primitive sequence:

1. The first sequence is initiated through a client application (for example, an inquiry for a service), which starts with a S-MethodInvoke.req operation to the server application within a session.
2. The second sequence is a response from the server to confirm to WSP in the client stack that the invoke message (the inquiry) has been received by the server.
3. The third sequence returns the result request (reply to the inquiry) from the server to the client.
4. The fourth sequence confirms the receipt to the server that the reply was received correctly.

18.9.5 Wireless Session Protocol (WSP)

WSP, defined by the OMA WAP-230-WSP-20010705-a specification, establishes a reliable session between the client and the server and releases that session in an orderly manner. An illustration of a client/server session is in Figure 18-13 on page 686. WSP session establishment agrees on a common level of protocol functionality using the capability of negotiation. WSP exchanges

content between the client and server using compact encoding, and also controls communication interrupt. Communication interrupt happens with change of a bearer, such as SMS (see Figure 18-3 on page 661).

WSP defines two protocols:

- Connection-mode session services over a transaction service

  This mode is used for long-lived connections. A session state is maintained. There is reliability for data sent over a connection-mode session.

- Non-confirmed, connection-less services over a datagram transport service

  This service is suitable when applications do not need reliable delivery of data and do not care about confirmation. It can be used without actually having established a session.

WSP provides semantics and mechanisms based on Hypertext Transport Protocol (HTTP) 1.1 and enhancements for wireless networks and its WAP terminals. Such enhancements include things such as long-lived sessions and data pushing, capability negotiation, and the ability to suspend and resume a session.

Basic functionality

Content headers are used to define content type, character set encoding, languages, and so on. Compact binary encoding is defined for the well-known headers to reduce protocol inefficiencies. A compact data format is supported that provides content headers for each component within the composite data object. This is a semantically equivalent binary form of the MIME-multipart/multimixed format used by HTTP 1.1.

As part of the session establishment process, request and response headers that remain constant over the life of the session can be exchanged between the service users in the client and the server. WSP passes through client and server session headers, as well as request and response headers, without additions or removals.

The life cycle of a WSP session is not tied to the underlying transport protocol. A session re-establishment protocol has been defined that allows sessions to be suspended and resumed without the processing costs of initial establishment. This allows a session to be suspended while idle to release network resources or save battery power. The session can be resumed over a different bearer network.

Layer-to-layer communication

Communications between layers and between entities within the session layer are accomplished through service primitives. They represent the logical

exchange of information and control between the session layer and adjacent layers.

Service primitives consist of commands and their respective response associated with the service provided and parameters. For example:

```
X-service.type (parameter)
```

Where X designates the layer providing the service; for WSP, it is S. The service types are the same as those used for WDP, illustrated in Table 18-1 on page 674.

When using service primitives, additional parameters are possible. They describe certain types of parameters. For example:

Addresses They describe client and server addresses to establish the session.

Headers and body They describe the HTTP entity-body. The headers distinguish between requests and responses sent from client to the server or reverse. The body contains the content of the message.

Capabilities These are service facilities, for example:

- Largest transaction data unit
- Set of code page names
- Maximum of outstanding requests

The capabilities can be negotiated between the client and the server.

Push identifier Indicates that the received message is a push transaction of the session that is pending on the service interface.

Reason The service provider uses the reason type to report the cause of a particular state of a primitive. Valid reasons are:

PROTOERR The rules of the protocol prevented the peer from performing the operation in its current state. For example, the used PDU was not allowed.

DISCONNECT The session was disconnected while the operation was still in progress.

SUSPEND The session was suspended while the operation was still in progress.

RESUME	The session was resumed while the operation was still in progress.
CONGESTION	The peer implementation could not process the request due to lack of resources.
CONNECTERR	An error prevented session creation.
MRUEXCEEDED	The SDU size in a request was larger than the maximum receive unit negotiated with the peer.
MOREXCEEDED	The negotiated upper limit on the number of simultaneously outstanding method or push requests was exceeded.
PEERREQ	The service peer requested the operation to be aborted.
NETERR	An underlying network error prevented completion of a request.
USERREQ	An action of the local service user was the cause of the indication.

Service primitive sample

The behavior of service primitives is illustrated using a time sequence chart (Figure 18-12).

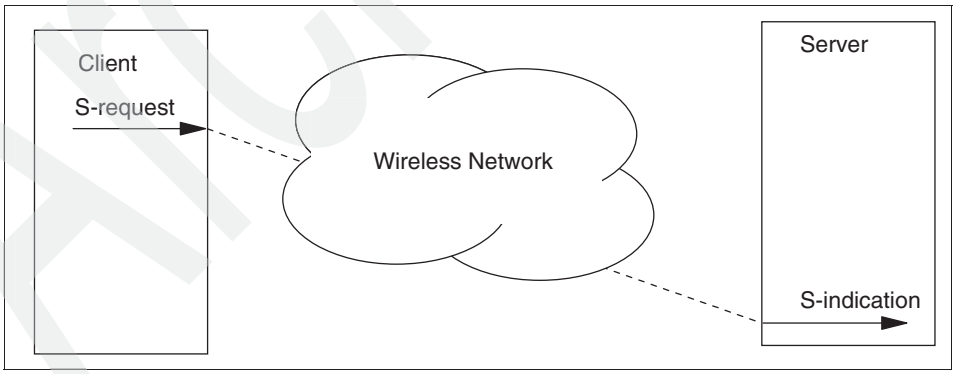


Figure 18-12 Non-confirmed service

This figure depicts a simple non-confirmed service. The client invokes an S-request primitive, which results in an S-indication primitive in the server (WSP

peer layer). The dashed line represents the propagation through the provider over a period of time.

Session services and operations

WSP is designed to function above transaction services, and also directly on datagram services without using the WTP layer. This means WSP can communicate with WDP (see 18.9.1, “Wireless Datagram Protocol (WDP)” on page 672) or with WTP (see 18.9.4, “Wireless Transaction Protocol (WTP)” on page 679), depending on the architecture of the application. Figure 18-13 shows two sessions from a WAP client to a WAP server.

One WAE application (represented by the dashed line) uses services from the session layer (WSP), the transport layer (WDP), and the network layer in the WAP client when it sends a message over the wireless network to the WAP proxy. In the WAP proxy, the equivalent peer layers are also used to reach the receiving WAE.

The other WAE application (represented by the solid line) uses additional services from the transaction layer (WTP). This is because an additional class of services is required for particular transactions (see 18.9.4, “Wireless Transaction Protocol (WTP)” on page 679).

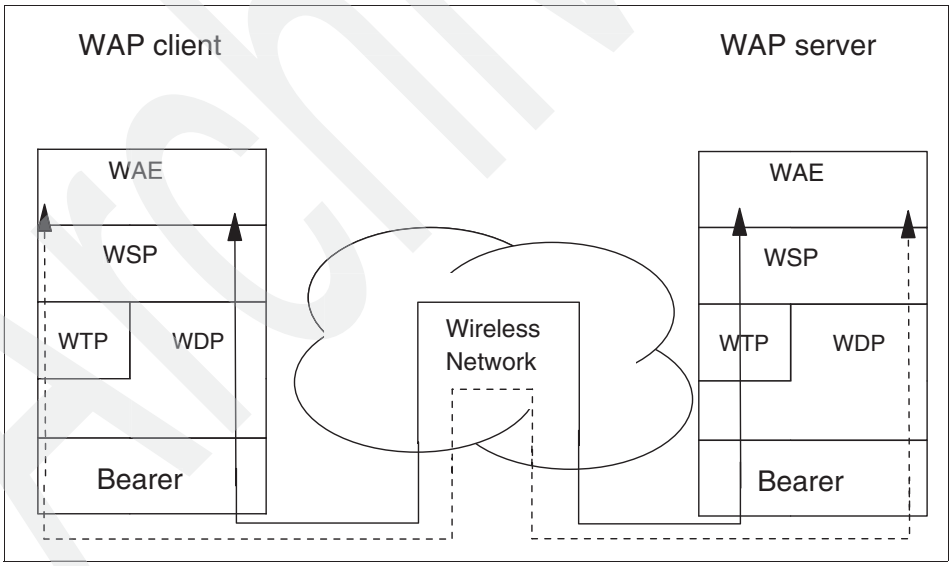


Figure 18-13 Client/server connection flow showing two WSP sessions

WSP connection-mode

Connection-mode session services provide two standard facilities:

- Session management facility

  The session management facility allows a client to connect to a server based on an agreement of facilities and options to use. During session establishment, the client can also exchange attribute information for the duration of the session. Although session establishment can only be done by the client, sessions can be terminated by both. The peer is invoked if one side tries to terminate the session. The user agent is also notified.

- Exception reporting facility

  The exception reporting facility allows the service provider to notify the user about events during the session.

There are some other facilities that are controlled through negotiation capabilities during session establishment. These are:

- Method invocation facility

  The method invocation facility permits the client to ask the server to execute an operation and return the result to the client. These operations can be compared with HTTP methods such as GET, PUT, and so on (see RFC 2616), or user-defined operations that fit into the same request/reply or transaction pattern. The service users are notified about completion of the operation whether it was successful or it failed.

- Push facility

  The push facility allows the server to send unsolicited information to the client. The sent information is non-confirmed. Delivery might be not reliable.

- Confirmed push facility

  The confirmed push facility is the same function as described in the previous paragraph. However, this unsolicited information forces an acknowledgment from the client to the server.

- Session resume facility

  The session resume facility allows both peers to suspend the session. The current session state is preserved. Further communication through this session is not possible until the client resumes the session. This function can also be used to switch the session to an alternate bearer.

WSP connection-mode: Negotiation capability

Information that is related to the operation of a session service provider is handled through capabilities. Capability negotiation is used between peers to agree on an acceptable level of service.

The peer who starts the negotiation process is the initiator and the other peer is the responder. The initiator only proposes a set of capabilities. The responder agrees or rejects the proposal.

Negotiations are about capabilities, such as:

- A list of alternate addresses for the same requested service
- An agreement on size of the largest size of transaction data
- Header code pages
- Maximum outstanding method requests
- Maximum outstanding push requests

WSP connection-mode: Operations

Figure 18-14 on page 689 shows a sample of a successful session establishment process through the client with subsequent invocation of an action in a server. Several service primitives are used on the client and the server side:

- S-Connect
  Used to initiate the session establishment and to notify the successful execution. Several parameters are provided, such as client address (session originator), server address (session target), client and server headers (application-level parameters to indicate that request and response headers of both partners are used throughout the session), and requested and negotiated capabilities for the duration of the session (for example, list of alias-addresses, client send data unit size, server send data unit size, maximum outstanding method requests, and maximum outstanding push requests).
- S-MethodInvoke
  Used to request an operation to be executed by a server. This service primitive can be used only together with S-MethodResult, which returns the result from the server to the client after the execution. The following parameters are valid: client and server transaction ID (to distinguish between several pending transactions over the same session), method (to tell which operation has to be used, either an HTTP method such as GET or PUT, or one of the extension methods established during capability negotiation), request Uniform Resource Identifier (URI) (to determine to which entity the operation applies), request headers (equivalent to HTTP headers), and request body (to contain the data associated with the request).

- S-MethodResult
  This service primitive returns the response to an operation request. It can be invoked only after a preceding S-MethodInvoke has occurred. The following parameters can be used for this service primitive: client and server transaction ID (distinguishes between pending transactions), *status* (tells, through the equivalent of the HTTP status code, RFC 2616, the state of the requested operation invoked by S-MethodInvoke through the client), response headers (as described under S-MethodeInvoke, the response body, the data associated with the response or, if the status indicates an error, contains further detailed error information), and acknowledgment headers (can be used to return some information back to the server).
- S-Disconnect
  Used to disconnect a session, or to notify a user that a session could not be established. Prior to issuing this, any incomplete transactions must be aborted.

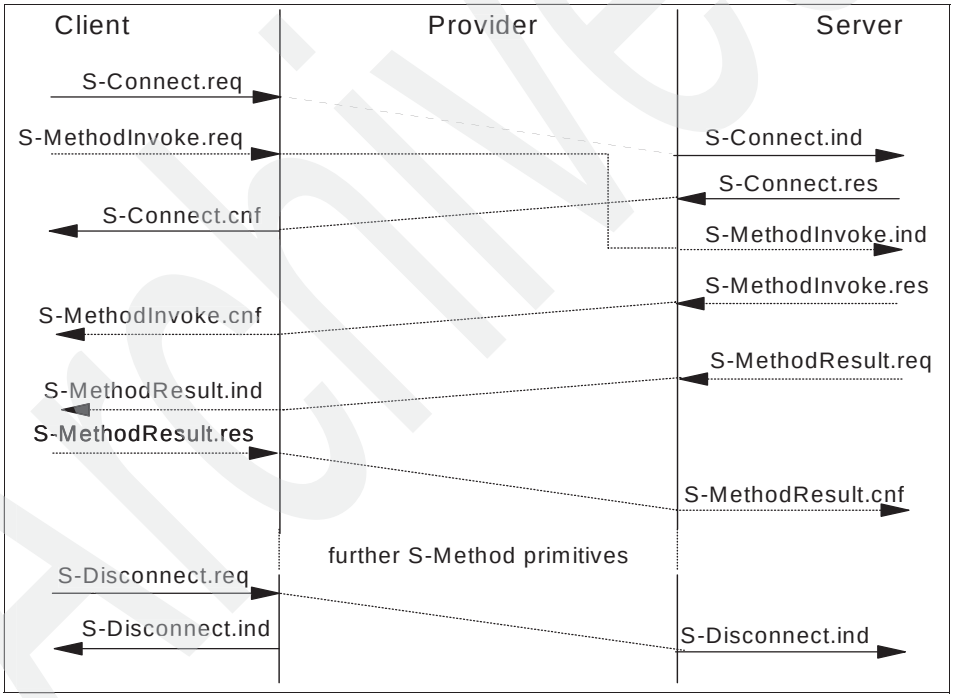


Figure 18-14 Normal session flow: Establishment, actions, and disconnection

This sequence of events can be explained as follows:

1. The client's session layer starts with a S-Connect.req request. The server's session layer notifies its upper layer (application layer) through an S-Connect.ind indication that a connection request is received.

2. The server uses the S-Connect.res response primitive to acknowledge reception of the indication primitive. The client's session layer uses the confirm primitive to report that the activity (S-Connect.req) has been completed successfully.
3. During the session initiation process, the client's session layer can start immediately with sending S-MethodInvoke.req requests asking for an execution on the server's machine. This request indicates that the upper layer has to execute an operation on the server's side.
4. The server sends a S-MethodInvoke.res response to the client confirming that the previous S-MethodInvoke.req request was completed successfully. The client is informed through S-Method.cnf.
5. After the server executes the operation, it sends the result to the client through S-MethodResult.req. The client indicates the upper layer (the application layer) through S-Method.Result.ind the receipt of the returned data.
6. Because the session partners have negotiated confirmation of all requests (***.req), the client confirms the S-MethodResult.req request with an S-MethodResult.res. response.
7. Further requests and responses can use this session.
8. The active session can by terminated in this example by issuing a S-Disconnect.req request to the server through the client. The server sends a S-Diconnect.ind to the session services in the session layer, which finishes the session. Session services on the client side will be informed to take down the session through S-Diconnect.ind.

Because the session layer does not provide any sequencing between multiple overlapping method invocations, the indications might be delivered in a different order than the sent request. The same is valid also for responses and confirmations, as well as for the corresponding S-MethodResult primitives. The application has to handle the sequencing.

There are some other session service primitives that are not used in the samples of the WSP connection-mode. These are:

S-Suspend	Suspends a session.
S-Resume	Resumes a session that was suspended.
S-Exception	Reports events.
S-MethodAbort	Aborts an operation request that is not yet complete.
S-Push	Sends unsolicited information from the server.
S-ConfirmedPush	Sends unsolicited information from the server; however, the client has to confirm this information.

S-PushAbort Rejects a push operation for a confirmed push.

WSP connection-mode using Wireless Transaction Protocol

This section describes the operation of when WSP sessions are run over Wireless Transaction Protocol (WTP) services (see 18.9.4, “Wireless Transaction Protocol (WTP)” on page 679).

The advantage of using WTP transaction within WSP sessions is comparable to TCP. WTP provides reliable transmissions, selectively defined re-transmission of lost packets, segmentation, and reassembly of large messages and controlled data flow between sender and receiver.

Another advantage is the use of WTP transaction classes defined in “WTP classes of operation” on page 679. Table 18-3 shows how WSP uses these three transaction classes.

Table 18-3 WTP facilities and transaction classes

WSP facility	WTP transaction classes
Session Management	Class 0 and Class 2
Method Invocation	Class 2
Session Resume	Class 0 and Class 2
Push	Class 0
Confirmed Push	Class 1

In order to show the cooperation between session services and transaction services, some diagrams are added. These are samples of different transaction services used by session services.

Archived

Normal session establishment

Figure 18-15 shows the normal session establishment process.

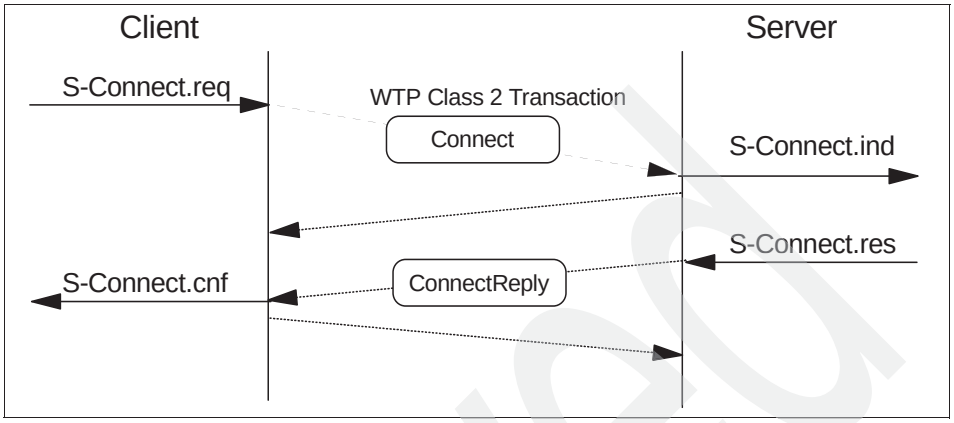


Figure 18-15 WSP WTP normal session establishment

Normal session termination

Figure 18-16 shows the normal session termination process.

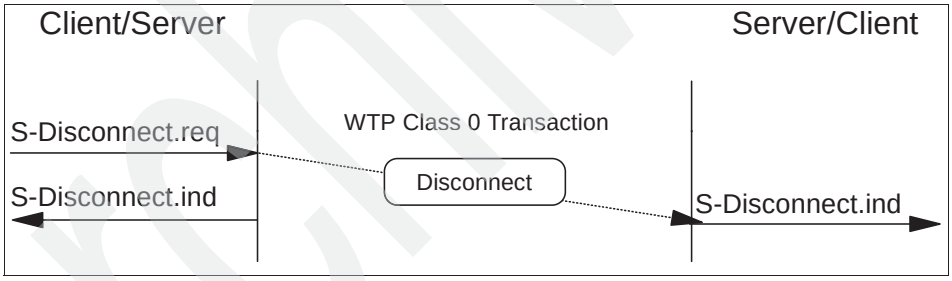


Figure 18-16 WSP WTP normal session termination

Archived

Normal session suspend and resume

Figure 18-17 shows the normal session suspend and resume process

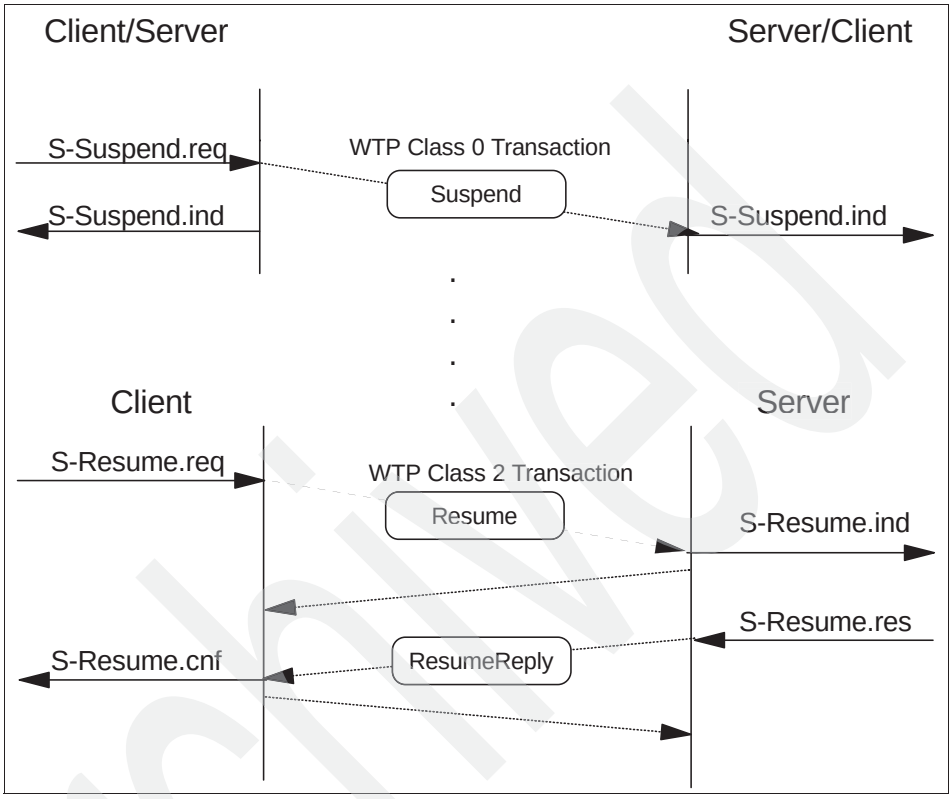


Figure 18-17 WSP WTP normal session suspend and resume

Archived

Normal method invocation

Figure 18-18 shows the normal method invocation process.

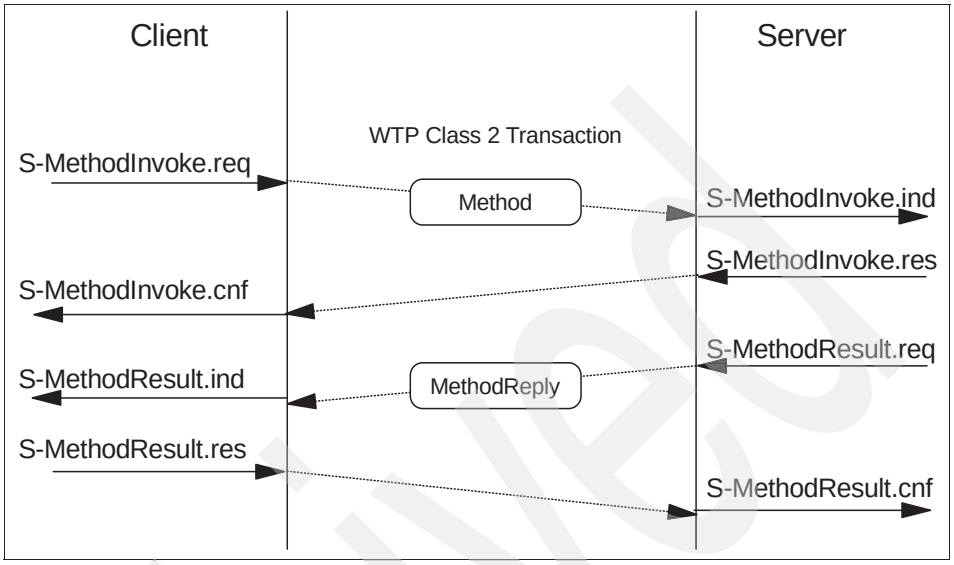


Figure 18-18 WSP-WTP normal method invocation

WSP connection-less

The connection-less session service provides non-confirmed facilities. They can be used to exchange content entities between layers. Like the connection-mode service, the connection-less service is asymmetric, which means no sequencing of messages.

This service is used mainly for the push facility.

The following service primitives are defined:

- S-Unit-MethodInvoke
- S-Unit-MethodResult
- S-Unit-Push

Event processing

Sessions are distinguished by an unique session identifier, which is valid during the duration of the session. Each session is associated with a peer quadruplet address. This quadruplet consists of:

- Client address
- Client port

- Server address
- Server port

Incoming transactions are assigned to a particular session based on the peer address quadruplet. This kind of session addressing scheme allows one session to be bound to a peer address quadruplet at a time.

18.9.6 Wireless profiled HTTP (W-HTTP)

In the original version of WAP (WAP1), mobile clients did not have HTTP functionality. As such, applications on a client wanting to obtain HTTP data were forced to use WML and a WAP proxy gateway. WML was used to make HTTP requests, and the proxy gateway translated the WML into HTTP through a series of encoders. This request was then sent to an origin server, and the resulting response then was translated by the proxy gateway back into WML and returned to the mobile client. This is illustrated in Figure 18-19.

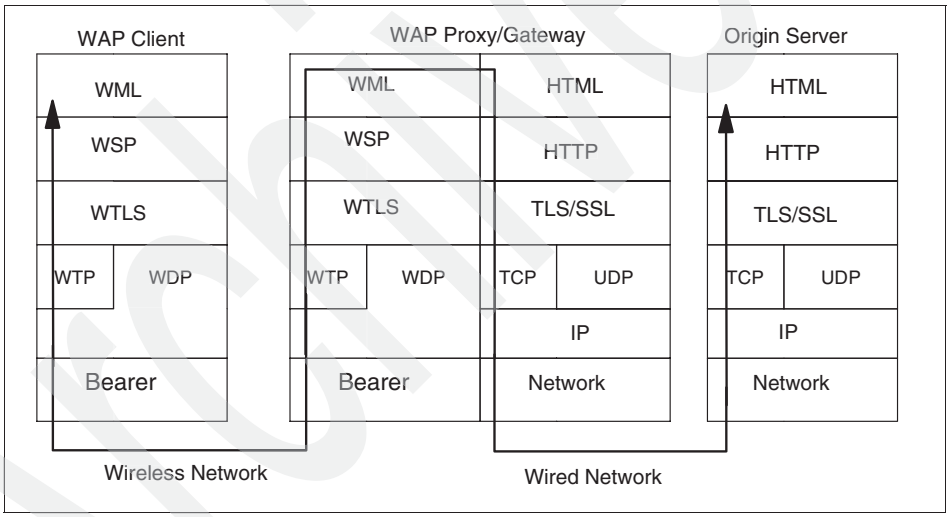


Figure 18-19 HTTP requests using WAP1 and a proxy/gateway

However, as the Open Mobile Alliance has continued to evolve the WAP standards such that more readily accommodate existing Internet protocols, a Wireless profiled HTTP (W-HTTP) standard has been designed in the OMA WAP-229-HTTP-20010329-a specification. Using W-HTTP, the request can exist over a split-TCP or end-to-end TCP connection (see 18.9.2, “Wireless Profiled Transmission Control Protocol (WP-TCP)” on page 674). Therefore, if using an end-to-end connection, the only intermediary needed is a device to transfer the data from the wireless network to the wired network. This can still be a proxy, or it can simply be an IP router, as illustrated in Figure 18-20 on page 696.

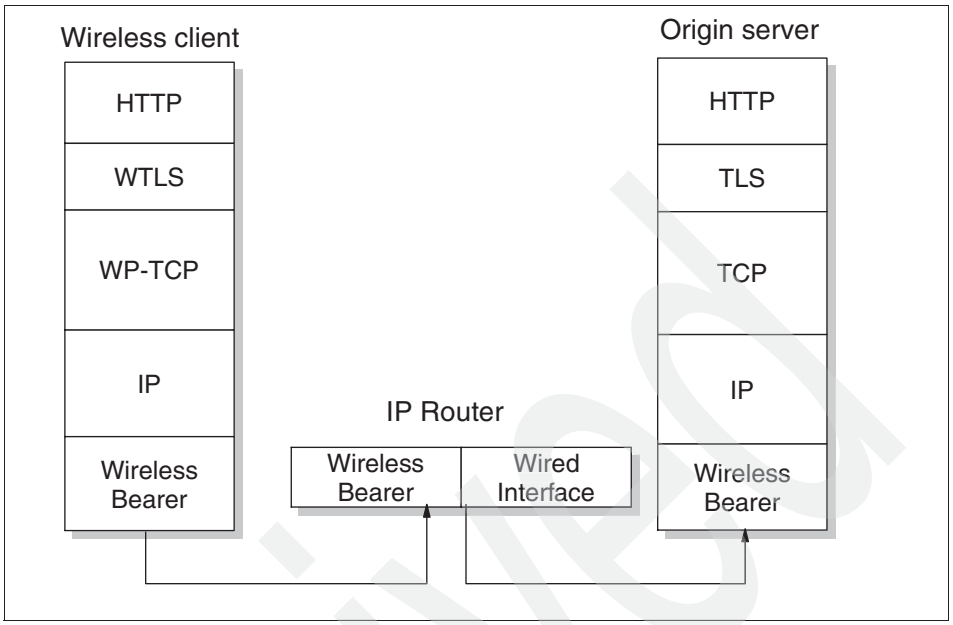


Figure 18-20 HTTP requests using WAP2 and an IP router

18.10 Wireless security

This section briefly outlines Wireless Transport Layer Security, its goals, and how it manages its connections and gives a brief protocol overview. In addition, this section provides a brief overview of the Wireless Identity Module.

18.10.1 Wireless Transport Layer Security (WTLS)

Many applications on the Web today require a secure connection between a client and the application server. WTLS is the security protocol, defined by the OMA WAP-261-WTLS-20010406-a specification, to ensure secure transactions on WAP terminals.

Note: WAP-261-WTLS-20010406-a is only the original specification. It has been updated by specifications WAP-261_100-WTLS-20010926-a, WAP-261_101-WTLS-20011027-a, and WAP-261_102-WTLS-20011027-a.

WTLS is based upon the industry-standard Transport Layer Security (TLS) protocol, but has been optimized for use over narrow-band communication channels. It ensures data integrity, privacy, authentication, and denial-of-service

protection. For Web applications that employ standard Internet security techniques with TLS, the WAP gateway automatically and transparently manages wireless security with minimal processing costs. It provides end-to-end security and application-level security. This includes security facilities for encrypting and decrypting, strong authentication, integrity, and key management. WTLS complies with regulations about the use of cryptographic algorithms and key lengths in different countries.

WTLS employs special adapted mechanisms for the wireless environment, for example, long existing secure sessions, optimized handshake procedures for the wireless network, and simple data reliability for operation over datagram bearers.

Figure 18-21 shows the location of WTLS in the WAP architecture model and the internal WTLS architecture with its different WTLS protocols.

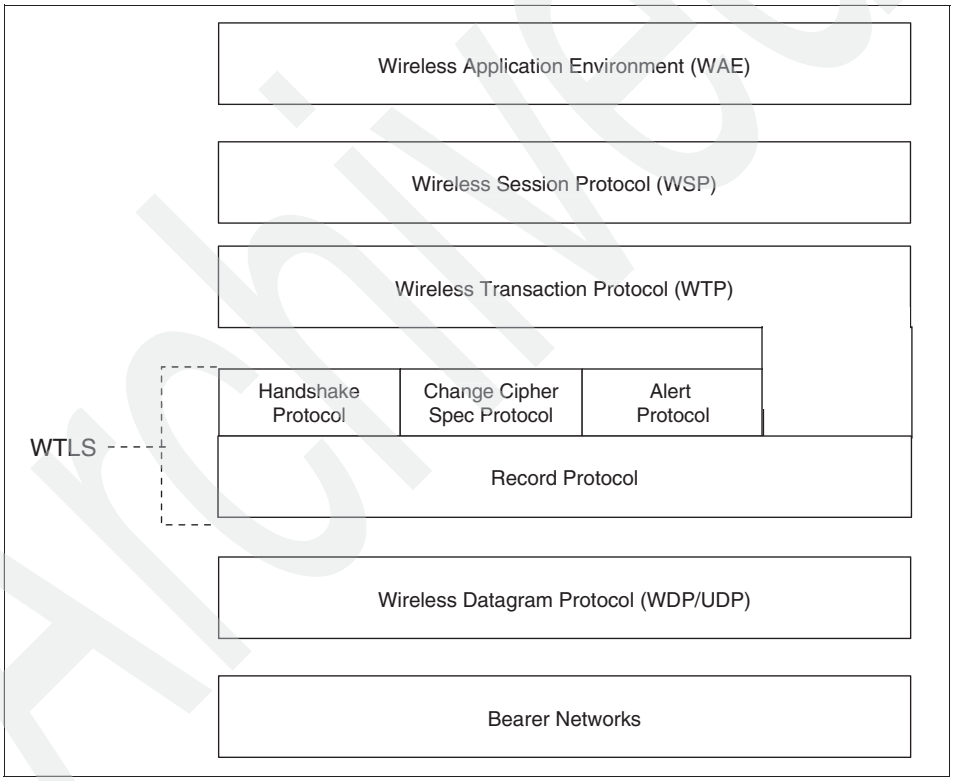


Figure 18-21 WTLS within the WAP architecture and WTLS components

Archive

WTLS layer goals

The primary goal is to provide security between the client and server in terms of:

Privacy	Data sent is not presented in clear text to make it difficult for another network user to look at the data. This is done through encrypting the data stream.
Data integrity	If a network user were able to change the sent data, it is detected by the client or server that sent the data. This is done through message digests.
Authentication	A network partner can be sure that he or she is connected with the true desired partner and not with someone else who pretends to be it. This is done through digital certificates.
Denial of service	Rejecting and detecting data that is not successfully verified. This is done through a WTLS function.

WTLS connection management

WTLS connection management provides a secure communication between the client and server. Several steps are needed within the connection establishment process through negotiations to agree to security parameters between the client and server. This is similar to a secure connection establishment process through Secure Socket Layer (SSL) (22.7, “Secure Sockets Layer (SSL)” on page 854).

Security parameters for the secure connection establishment process can include, for example, cryptographic algorithms, key exchange procedures, and authentication.

Service primitives provide support to initiate this secure connection. The following service primitives are available:

- SEC-Create: Initiates a secure connection with the following optional parameters:
  - Client certificates
  - Key exchange suites
  - Cipher suites
  - Compression methods
  - Key refreshing rules
  - Session ID

- SEC-Exchange: Performs public key authentication or key exchange with a client. Additional information about public keys is in OMA specifications WAP-217-WPKI-20010424-a, WAP-217_103-WPKI-20011102-a, and WAP-217_105-WPKI-20020816-a.
- SEC-Commit: Initiated when the handshake is completed and either peer requests to switch to the agreed connection state.
- SEC-Terminate: Terminates the connection.
- SEC-Exception: Informs the other partner about warning level alerts.
- SEC-Create-Request: The server requests the client to initiate a new handshake.

Protocol overview

As shown in Figure 18-21 on page 697, WTLS consists of four protocol components:

- The record protocol
- The handshake protocol
- The alert protocol
- The change CipherSpec protocol

Record protocol

The record protocol is the interface to the upper layer (transaction or session layer) and to the lower layer (transport layer). It receives messages from the upper layer to be transmitted, optionally compresses the data, applies a message authentication code (MAC), encrypts the message, and then transmits the message. Conversely, received data is decrypted, verified, decompressed, and delivered to a higher layer of the client. The remaining four protocols cooperate very closely with the record protocol in achieving these steps.

Handshake protocol

This protocol consists of three subprotocols that allow peers to agree on security parameters for the record layer. The handshake protocol is responsible for the negotiation process between the client and server and is employed when initiating WTLS. These parameters are negotiated during the handshake:

Session identifier	Identifies an active and resumeable secure session.
Protocol version	WTLS protocol version number.
Peer certificate	Certificate of the peer.
Compression method	The algorithm used to compress data prior to encryption.

CipherSpec	Specifies the bulk data encryption algorithm (such as null, RC5, DES, and so on) and MAC algorithm (such as SHA-1). It also defines cryptographic attributes, such as the MAC size.
Master secret	20-byte secret shared between client and server.
Sequence number mode	Sequence numbering scheme used in this secure connection.
Key refresh	Defines how often some connection state values (encryption key, MAC secret, and IV) calculations are performed.
Is resumeable	Flag indicating whether the secure session can be used to initiate new secure connection.

Four phases are used in the handshake protocol exchange, prior to sending application data:

1. First phase: During this phase, the connection is established and security capabilities are negotiated.
2. Second phase: During this phase, server authentication and key exchange occurs.
3. Third phase: During this phase, client authentication and key exchange occurs.
4. Fourth phase: Completion of the secure connection establishment.

Change CipherSpec protocol

The change CipherSpec is sent by the client or server to notify the other partner that subsequent records will be sent under the newly negotiated CipherSpec and keys. This message is sent during the handshake after the security parameters have been agreed on, but before the verifying finished message is sent.

Alert protocol

Alert messages contain information to a peer about an event occurring on a secure connection. Included in the alert is information about the severity of the message and a description of the alert. Alerts fall into two categories, Closure and Error. Closure alerts provide information about why a secure connection was closed or could not be opened, while error alerts provide details about any faults that might occur while a secure connection is in progress. Details about these alerts are on the OMA Web site in the WAP-219-TLS-20010411-a specification.

Wireless TLS tunnelling

Although one of the benefits achieved by implementing WAP2 is that a direct connection can be established between a WAP client and server, the WAP2

architecture still allows proxies to exist between client and servers. In such a scenario, the process of setting up end-to-end TLS capabilities becomes more complicated. The OMA WAP-219-TLS-20010411-a specification addresses this by defining the process of creating a TLS tunnel across a proxy.

In such a scenario, the proxy server acts only as a transport layer data relay, performing no processing on messages passed between the server and client, nor passing the message up to higher layers in the architecture. This is illustrated in Figure 18-22.

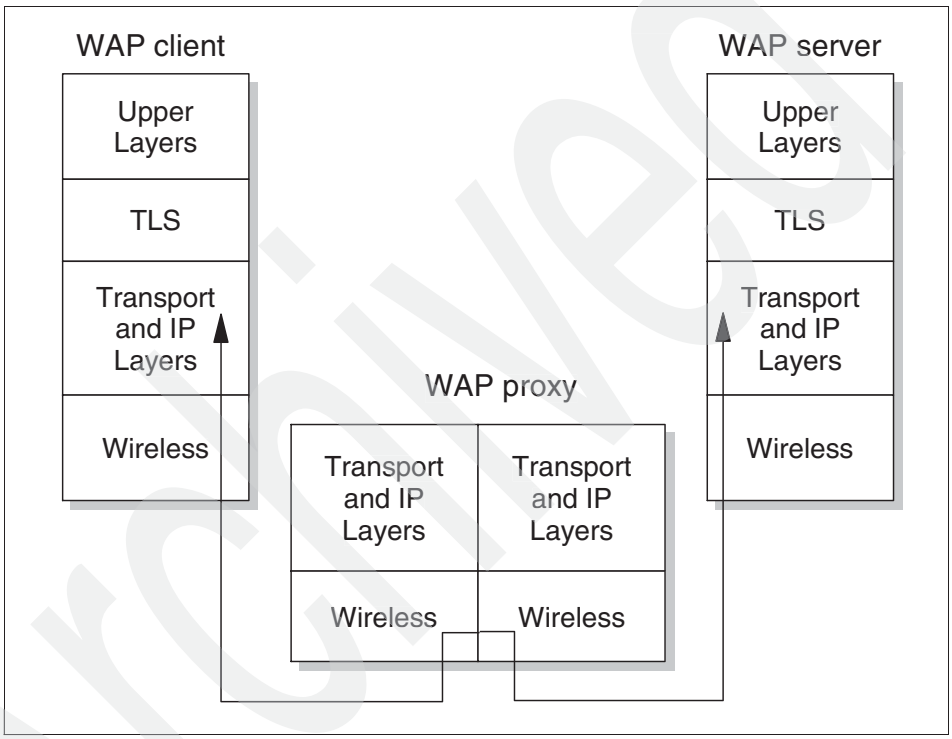


Figure 18-22 An example of TLS tunnelling

18.10.2 Wireless Identity Module (WIM)

In implementing WTLS, it is important to have a tamper-proof device that stores secret information (such as keys and certificates) and can also perform other security functions (such as cryptology). Such a device increases the difficulty faced by a malicious user attempting to gain access to such information. In order to achieve this functionality, WAP2 architecture defines the Wireless Identity Module (WIM). This typically exists as a smart card within the mobile device. Defined originally by the OMA WAP-260-WIM-20010712-a specification, the

WIM is used to store permanent private keys and use such keys in functions such as:

- Signing operations
- Key exchange operations
- Cryptographic operations
- Securing long-living WTLS sessions

These operations are outside the realm of networking, and thus warrant no further discussion. However, note that both WTLS layer and the application layer can invoke WIM functions.

18.11 Wireless Telephony Application (WTA)

WTA provides tools for building telephony applications. It is primarily designed for network operators, carriers, and equipment vendors. Network security and reliability is a major consideration. Extensions can be added to the standard WML/WMLScript browser to support an additional WTA Application Programming Interface (WTAI).

Because WTA is beyond the TCP/IP scope, it does not warrant further discussion. However, additional information is on the OMA Web site in the following specifications:

- WAP-266-WTA-20010908-a – Wireless Telephony Application Specification
- WAP-268-WTAI-20010908-a – Wireless Telephony Application Interface Specification
- WAP-255-WTAIGSM-20010908-a – WTAI, GSM Specific Addendum
- WAP-269-WTAIIS136-20010908-a – WTAI, IS-136 Specific Addendum
- WAP-270-WTAIPDC-20010908-a – WTAI, PDC Specific Addendum
- WAP-228-WTAIIS95-20010908-a – WTAI, IS95 Specific Addendum

18.12 RFCs relevant to this chapter

The following RFCs provide detailed information about the TCP/IP protocol suite and architectures presented throughout this chapter:

- RFC 0792 – Internet Control Message Protocol (September 1981)
- RFC 0793 – Transmission Control Protocol (September 1981)

- RFC 1122 – Requirements for Internet Hosts - Communication Layers (October 1989)
- RFC 1191 – Path MTU Discovery (November 1990)
- RFC 1323 – TCP Extensions for High Performance (May 1992)
- RFC 1981 – Path MTU Discovery for IP version 6 (August 1996)
- RFC 2018 – TCP Selective Acknowledgement Options (October 1996)
- RFC 2414 – Increasing TCP's Initial Window (September 1998)
- RFC 2463 – Internet Control Message Protocol (ICMPv6) (December 199)
- RFC 2481 – A Proposal to add Explicit Congestion Notification (ECN) to IP (January 1999)
- RFC 2581 – TCP Congestion Control (April 1999)
- RFC 2616 – Hypertext Transfer Protocol -- HTTP/1.1 (June 1999)
- RFC 3390 – Increasing TCP's Initial Window (October 2002)
- RFC 4443 – Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) (March 2006)

18.13 Specifications relevant to this chapter

The following Open Mobile Alliance specifications provide detailed information about the Wireless Application Protocol implementations and architectures presented throughout this chapter, and are available at:

`http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html`

- WAP-248-UAProf-20011020-a – User Agent Profiling Specification (October 2001)
- WAP-236-WAESpec-20020207-a – Wireless Application Environment Specification (February 2007)
- WAP-237-WAEMT-20010515-a – WAP Media Types Specification (May 2001)
- WAP-277-XHTMLMP-20011029-a – XHTML Mobile Profile Specification (October 2001)
- WAP-225-TCP-20010331-a – Wireless profiled TCP Specification (March 2001)
- WAP-229-HTTP-20010329-a – Wireless profiled HTTP Specification (March 2001)

Archive

- WAP-229_001-HTTP-20011031-a – Wireless profiled HTTP SIN 001 (October 2001)
- WAP-259-WDP-20010614-a – Wireless Datagram Protocol Specification (June 2001)
- WAP-202-WCMP-20010624-a – Wireless Control Message Protocol Specification (June 2001)
- WAP-224-WTP-20010710-a – Wireless Transaction Protocol Specification (July 2001)
- WAP-230-WSP-20010705-a – Wireless Session Protocol Specification (July 2001)
- WAP-187-TransportE2ESec-20010628-a – End-to-end Transport Layer Security Specification (June 2001)
- WAP-187_101-TransportE2ESec-20011009-a – End-to-end Transport Layer Security SIN 101 (October 2001)
- WAP-260-WIM-20010712-a – Wireless Identity Module Specification (July 2001)
- WAP-260_100-WIM-20010725-a – Wireless Identity Module Specification (July 2001)
- WAP-260_101-WIM-20020107-a – Wireless Identity Module Specification (January 2002)
- WAP-261-WTLS-20010406-a – Wireless Transport Layer Security Specification (April 2001)
- WAP-261_100-WTLS-20010926-a – Wireless Transport Layer Security SIN 100 (September 2001)
- WAP-261_101-WTLS-20011027-a – Wireless Transport Layer Security SIN 101 (October 2001)
- WAP-261_102-WTLS-20011027-a – Wireless Transport Layer Security SIN 102 (October 2001)
- WAP-217-WPKI-20010424-a – WAP Public Key Infrastructure Specification (April 2001)
- WAP-217_103-WPKI-20011102-a – WAP Public Key Infrastructure SIN 103 (November 2001)
- WAP-217_105-WPKI-20020816-a – WAP Public Key Infrastructure SIN 105 (August 2002)
- WAP-219-TLS-20010411-a – WAP TLS Profile and Tunneling Specification (April 2001)

Archived

- WAP-219_100-TLS-20011029-a – WAP TLS Profile and Tunneling SIN 100 (October 2001)
- WAP-266-WTA-20010908-a – Wireless Telephony Application Specification (September 2001)
- WAP-268-WTAI-20010908-a – Wireless Telephony Application Interface Specification (September 2001)
- WAP-255-WTAIGSM-20010908-a – WTAI, GSM Specific Addendum (September 2001)
- WAP-269-WTAIIS136-20010908-a – WTAI, IS-136 Specific Addendum (September 2001)
- WAP-270-WTAIPDC-20010908-a – WTAI, PDC Specific Addendum (September 2001)
- WAP-228-WTAIIS95-20010908-a – WTAI, IS95 Specific Addendum (September 2001)

Archived

Archived

19

Presence over IP

This chapter provides an overview of presence, how the presence service operates, and the protocols associated with the presence. This chapter includes the following sections:

- Overview of the presence service
- Presence Information Data Format (PIDF)
- Binding to TCP
- Address resolution
- RFCs relevant to this chapter

Presence is a means for finding, retrieving, and subscribing to changes in the presence information (for example, “online” or “offline”) of other users. Instant messaging is a means for sending small, simple messages that are delivered immediately to online users. Instant messaging is discussed in further detail in RFC 2778, RFC 2779, and RFC 3860.

Presence is normally discussed with instant messaging because they are generally used in conjunction with each other. However, they are two distinct services and can function independently of each other. The remainder of this chapter focuses on the concept of a presence service, and not much on instant messaging services.

© Copyright IBM Corp. 1989-2006. All rights reserved.

The following terminology is used with presence services:

Principal	Human, program, or a collection of humans, programs, or both, that chooses to appear to the presence service as a single actor, distinct from all other principals.
Presentity	A presence entity is called a presentity. Each presentity provides presence information to a presence service. The presentity is not (usually) located within the presence service; the presence service only has a recent version of the presentity's presence information. The presentity initiates changes in the presence information, which is then distributed by the presence service.
Presence information	Consists of one or more presence tuples.
Presence tuple	Consists of elements associated with a presentity's presence information. for example, status and communication address.
Status	A distinguished part of the presence information of a presentity. Status has at least the mutually-exclusive values open and closed, which have meaning for the presentity being online or offline.
Open	A distinguished value of the status marker. The associated presentity is online. Contrast with closed.
Closed	A distinguished value of the status marker. The associated presentity is offline. Contrast with open.
Communication address	Consists of communication means and a contact address.
Communication means	Indicates a method whereby communication can take place. An instant message service is one example of a communication means.
Contact address	A specific point of contact through some communication means.
Other presence markup	Any additional information included in the presence information of a presentity.
Fetcher	A form of watcher that has asked the presence service for the presence information of one or more presentities.
Poller	A fetcher that requests presence information on a regular basis.

Subscriber	A form of watcher that has asked the presence service to notify it immediately of changes in the presence information of one or more presentities.
Subscription	The information kept by the presence service about a subscriber's request to be notified of changes in the presence information of one or more presentities.
Notification	A message sent from the presence service to a subscriber when there is a change in the presence information of some presentity of interest, as recorded in one or more subscriptions.
Watcher	Requests presence information about a presentity, or watcher information about a watcher, from the presence service. Special types of watcher are fetcher, poller, and subscriber.
Watcher information	Information about watchers that have received presence information about a particular presentity within a particular recent span of time. Watcher information is maintained by the presence service, which can choose to present it in the same form as presence information; that is, the service can make watchers look like a special form of presentity.
Access rule	Constraints on how a presence service makes presence information available to watchers. For each presentity's presence information, the applicable access rules are manipulated by the presence user agent of a principal that controls the presentity.
Visibility rules	Constraints on how a presence service makes watcher information available to watchers. For each watcher's watcher information, the applicable visibility rules are manipulated by the watcher user agent of a principal that controls the watcher.
Presence service	Accepts, stores, and distributes presence information. • Can require authentication of presentities and watchers. • Can have different authentication requirements for different presentities.

- Can have different authentication requirements for different watchers, and can also have different authentication requirements for different presentities being watched by a single watcher.
- Can have an internal structure involving multiple servers, proxies, or both. There can be complex patterns of redirection and proxying while retaining logical connectivity to a single presence service. Note that a presence service does not require having a distinct server.

  The service can be implemented as direct communication among presentity and watchers.
- Can have an internal structure involving other presence services, which can be independently accessible in their own right as well as being reachable through the initial presence service.

Presence protocol The interaction between presence service, presentities, and watchers. Presence information is carried by the presence protocol.

Presence user agent Means for a principal to manipulate zero or more presentities.

19.1 Overview of the presence service

RFC 2778 defines a model and terminology for describing systems that provide presence information, and RFC 2779 defines the requirements that a presence protocol must fulfill. The following discussion is based on RFC 2778 with added features from RFC 3863.

Archive

The presence service has two distinct sets of “clients.” One set of clients, called presentities, provides presence information to be stored and distributed by the presence service. The other set of clients, called watchers, receives presence information from the service. This is illustrated in Figure 19-1.

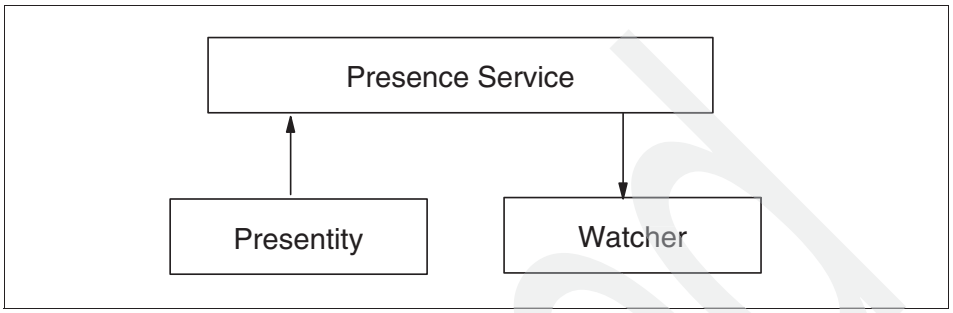


Figure 19-1 Overview of presence service

There are two kinds of watchers, called fetchers and subscribers. A fetcher simply requests the current value of some presentity’s presence information from the presence service. In contrast, a subscriber requests notification from the presence service of (future) changes in some presentity's presence information. A special kind of fetcher, called a poller, fetches information on a regular basis. Figure 19-2 illustrates the variety of watchers.

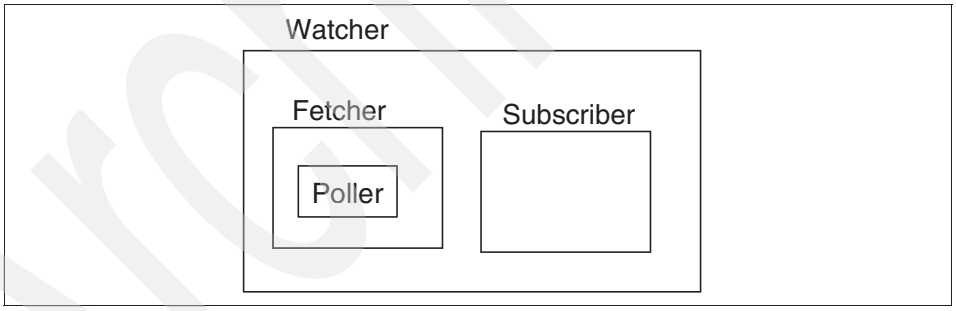


Figure 19-2 Watcher varieties

The presence service also has watcher information about watchers and their activities in terms of fetching or subscribing to presence information. The presence service can also distribute watcher information to some watchers, using the same mechanisms that are available for distributing presence information.

Archived

Changes to presence information are distributed to subscribers through notifications. Figure 19-3, Figure 19-4 and Figure 19-5 on page 713 show the flow of information as a piece of presence information is changed from P1 to P2.

As illustrated in Figure 19-3, P1 is changed to P2 on the presentity.

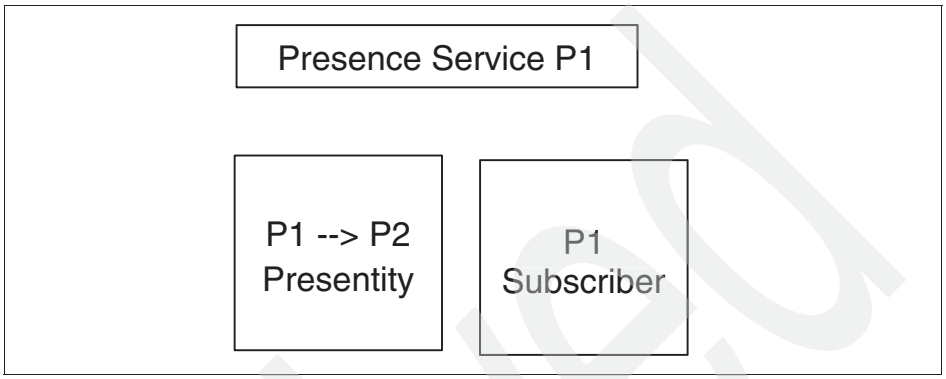


Figure 19-3 Notification step 1

As illustrated in Figure 19-4, the changed value (P2) is sent through a notification to the presence service, which then changes its stored value of P1 to P2.

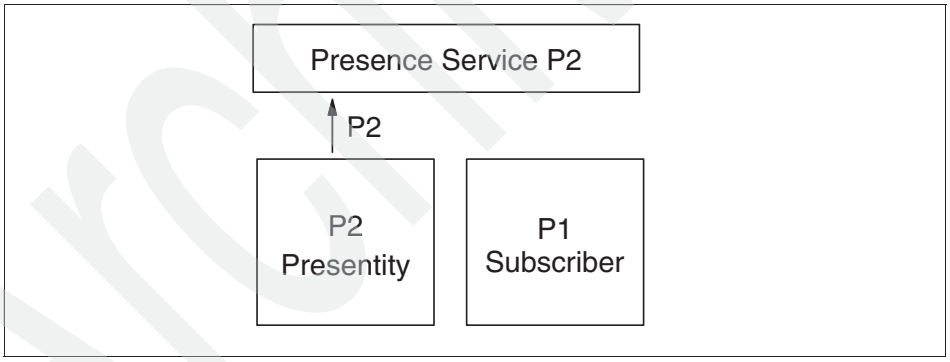


Figure 19-4 Notification step 2

Archived

As illustrated in Figure 19-5 The changed value (P2) is then sent to the subscriber of the presentity's presence information.

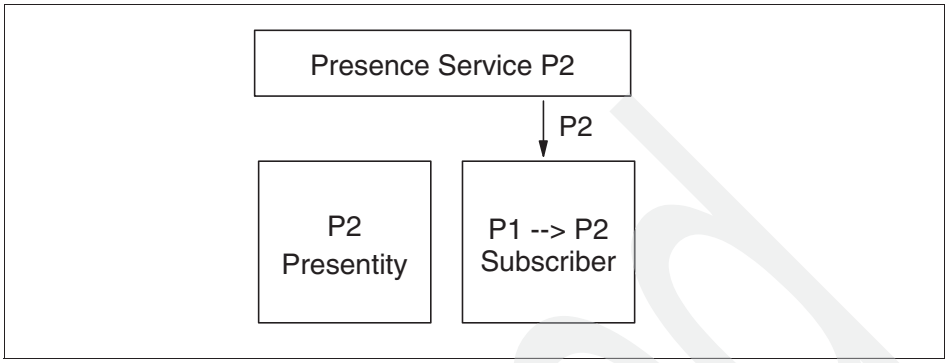


Figure 19-5 Notification step 3

A simple example of a presence system is a generic "buddy list" application. These applications typically expose the user's presence to others and make it possible to see the presence of others.

A presence system is illustrated in Figure 19-6.

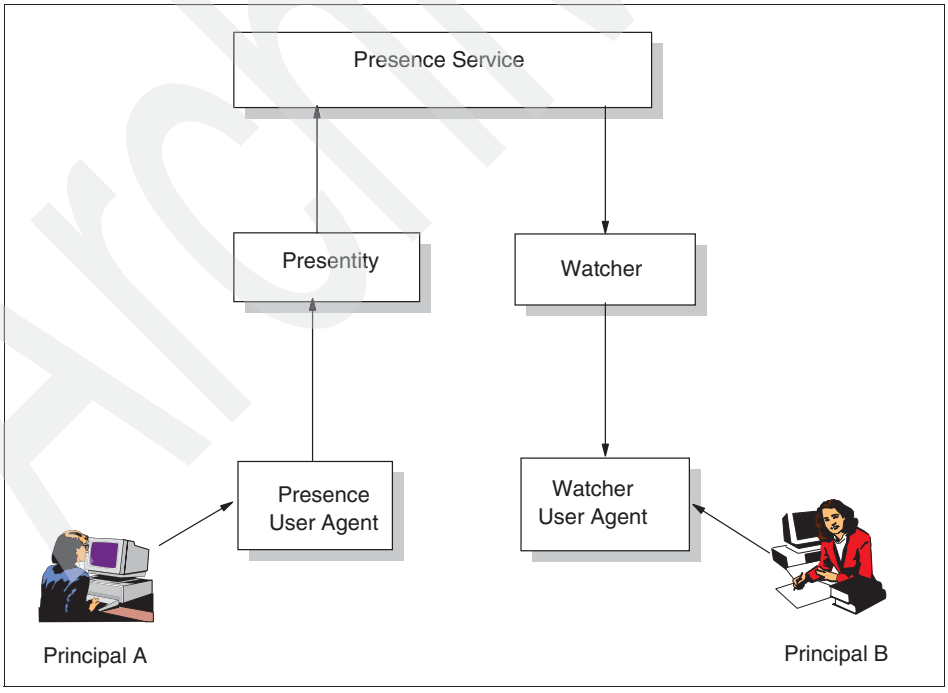


Figure 19-6 A presence system

19.2 Presence Information Data Format (PIDF)

RFC 3859 – Common Profile for Presence (CPP) defines common semantics and data formats for presence to facilitate the creation of gateways between presence services. This assists with the interoperability issues between presence services. RFC 3863 defines the Presence Information Data Format (PIDF) as a common presence data format for CPP-compliant presence protocols, allowing presence information to be transferred across CPP-compliant protocol boundaries without modification. This is because the PIDF encodes presence information in Extensible Markup Language (XML). The presence information is therefore hierarchically structured and fully extensible.

The PIDF is as follows:

- Presentity URL: This is the “pres” Uniform Resource Locator (URL) of the presentity.
- List of presence tuples, which contain the following elements:
  - Identifier: Token to identify this tuple within the presence information.
  - Status: Open/Closed or extension status values, or both.
  - Communication address: Communication means and contact address of this tuple (optional). A contact address can be an e-mail address or a telephone number. RFC 3953 discusses the Telephone Number Mapping (ENUM) service for presence.
  - Relative priority: Numerical value specifying the priority of this communication address (optional).
  - Time stamp: Time stamp of the change of this tuple (optional).
  - Human-readable comment: Free text memo about this tuple (optional).
  - Presentity human-readable comment: Free text memo about the presentity (optional).
  - Timed-status: Status of a presentity that is either no longer valid or covers some future time period. (This element is an addition to the PIDF and is discussed in RFC 4481.)

For further information regarding these elements, refer to RFC 3863 and RFC 4481 for the timed-status element specifically.

Rich Presence Information Data (RPID) format is discussed in RFC 4480, which includes other XML elements that extend the PIDF. These elements include:

- activities: Enumerates what the person is doing.
- class: An identifier that groups similar person elements, devices, or services.

- deviceID: Indicates that this device contributes to the service described by the tuple.
- mood: Indicates the mood of the person.
- place-is: Reports on the properties of the place the presentity is currently, such as the levels of light and noise.
- place-type: Reports the type of place the person is located, such as “classroom” or “home.”
- privacy: Distinguishes whether the communication service is likely to be observable by other parties.
- relationship: When a service is likely to reach a user besides the person associated with the presentity, the relationship indicates how that user relates to the person.
- service-class: Describes whether the service is delivered electronically, is a postal or delivery service, or describes in-person communications.
- sphere: Characterizes the overall current role of the presentity.
- status-icon: Depicts the current status of the person or service.
- time-offset: Quantifies the time zone the person is in, expressed as the number of minutes away from UTC.
- user-input: Records the user-input or usage state of the service or device, based on human user input.

RFC 4482 adds Contact Information for the PIDF (CIPID) elements to the PIDF that provide additional contact information about a presentity and its contacts. The elements include:

- card: Includes a URI pointing to a business card, for example, in LDAP Data Interchange Format (LDIF) (RFC 2849) or vCard (RFC 2426) format.
- display-name: Includes the name identifying the tuple or person that the presentity suggests should be shown by the watcher user interface.
- homepage: Provides a URI pointing to general information about the tuple or person, typically a Web home page.
- icon: Provides a URI pointing to an image (icon) representing the tuple or person.
- map: Provides a URI pointing to a map related to the tuple or person.
- sound: Provides a URI pointing to a sound related to the tuple or person. The sound object might also be used to indicate how to pronounce the presentity's name or even an MP3.

19.3 Presence protocols

The Extensible Messaging and Presence Protocol (XMPP) is an open Extensible Markup Language (XML) protocol for near-real-time messaging, presence, and request-response services and is discussed in RFC 3920 with extensions in RFC 3921.The basic syntax and semantics were developed originally in the Jabber open source community, mainly in 1999. In 2002, the XMPP WG was chartered with developing an adaptation of the Jabber protocol that would be suitable as an IETF instant messaging (IM) and presence technology. XMPP provides the basic instant messaging and presence functionality defined in RFC 2779. The following discussion is based on these RFCs.

Although XMPP is not wedded to any specific network architecture, to date it usually has been implemented through a client/server architecture wherein a client using XMPP accesses a server over a TCP connection, and servers also communicate with each other over TCP connections. The XMPP architecture is illustrated in Figure 19-7, where a single line (_) depicts communication through XMPP and a double line (=) depicts communication through a non-XMPP protocol.

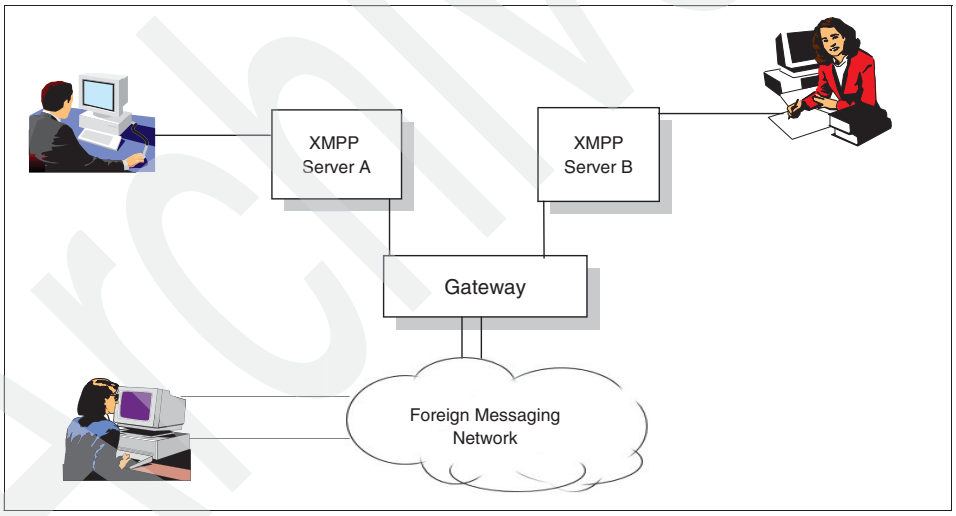


Figure 19-7 XMPP architecture

We describe each of the objects in the following sections.

XMPP server

An XMPP server acts as an intelligent abstraction layer for XMPP communications. Its primary responsibilities are:

- To manage connections from or sessions for other entities, in the form of XML streams to and from authorized clients, servers, and other entities
- To route appropriately-addressed XML stanzas among such entities over XML streams

An *XML stream* is a container for the exchange of XML elements between any two entities over a network and an *XML stanza* is a discrete semantic unit of structured information that is sent from one entity to another over an XML stream.

Most XMPP-compliant servers also assume responsibility for the storage of data that is used by clients (for example, contact lists for users of XMPP-based instant messaging and presence applications). In this case, the XML data is processed directly by the server itself on behalf of the client and is not routed to another entity.

XMPP client

Most clients connect directly to a server over a TCP connection and use XMPP to take full advantage of the functionality provided by a server and any associated services. Multiple resources (for example, devices or locations) can connect simultaneously to a server on behalf of each authorized client, with each resource differentiated by the resource of an XMPP address (for example, <node@domain/ home> versus <node@domain/work>) as defined under addressing scheme. The recommended port for connections between a client and a server is 5222, as registered with the IANA.

Non-XMPP client

This client does not use XMPP and uses a different protocol.

Gateway

The gateway is a special-purpose, server-side service whose primary function is to translate XMPP into the protocol used by a foreign (non-XMPP) system, as well as to translate the return data back into XMPP. Examples are gateways to e-mail (refer to 15.1, “Simple Mail Transfer Protocol” on page 556), Internet Relay Chat (IRC), Short Message Service (SMS), and enterprise instant messaging services such as AIM, ICQ, MSN® Messenger, and Yahoo! Instant Messenger.

Network

Because each server is identified by a network address and because server-to-server communications are a straightforward extension of the

client-to-server protocol, in practice, the system consists of a network of servers that inter-communicate. Therefore, for example, <juliet@example.com> is able to exchange messages, presence, other information with <romeo@example.net>. This pattern is familiar and from messaging protocols (such as SMTP) that make use of network addressing standards. Communications between any two servers are optional. If enabled, such communications occur over XML streams that are bound to TCP connections. The recommended port for connections between servers is 5269, as registered with the IANA.

19.3.1 Binding to TCP

Although there is no necessary coupling of an XML stream to a TCP connection (for example, two entities can connect to each other through another mechanism such as polling over HTTP), this specification addressed in RFC 3920 defines a binding of XMPP to TCP only. In the context of client-to-server communications, a server must allow a client to share a single TCP connection for XML stanzas sent from client to server and from server to client. In the context of server-to-server communications, a server must use one TCP connection for XML stanzas sent from the server to the peer and another TCP connection (initiated by the peer) for stanzas from the peer to the server, for a total of two TCP connections.

19.3.2 Address resolution

A client determines the IP address of an appropriate system running on a server by resolving the destination domain name, which is part of the identifier within the presence tuple, to either an intermediate relay system or a final target system. This is discussed further in RFC 3861.

19.4 RFCs relevant to this chapter

The following RFCs provide detailed information about the presence services and concepts presented throughout this chapter:

- RFC 2426 – vCard MIME Directory Profile (September 1998)
- RFC2778 – A Model for Presence and Instant Messaging (February 2000)
- RFC 2779 – Instant Messaging / Presence Protocol Requirements (February 2000)
- RFC 2849 – The LDAP Data Interchange Format (LDIF) - Technical Specification (June 2000)

- RFC 3859 – Common Profile for Presence (CPP) (August 2004)
- RFC 3861 – Address Resolution for Instant Messaging and Presence (August 2004)
- RFC 3863 – Presence Information Data Format (PIDF) (August 2004)
- RFC 3920 – Extensible Messaging and Presence Protocol (XMPP): Core (October 2004)
- RFC 3953 – Telephone Number Mapping (ENUM) Service Registration for Presence Services (January 2005)
- RFC 3921 – Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (October 2004)
- RFC 4480 – RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF) (July 2006)
- RFC 4481 – Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Status Information for Past and Future Time Intervals (July 2006)

Archived

Archived

Part 3

Advanced concepts and new technologies

As networks and networked resources continue to become more omnipresent in modern business models, the need for standards and interoperability between networks becomes a greater concern. Similar to this is the need for uninterrupted access to networked resources. However, prior to the convergence of networks and services to an IP model, some networks were designed solely with data transportation in mind. Other networks were designed with potential growth and scalability in mind. Additionally, there was no requirement to develop vertical services with IP awareness. As a result, there currently exist multiple sets of disparate networks and services unable to interoperate or take advantage of the advances offered by IP.

In an effort to resolve this, organizations have begun moving toward an implementation of networks and services over the IP model. By converging

© Copyright IBM Corp. 1989-2006. All rights reserved.

networks in such a manner, organizations can now provide IP connectivity, quality of service, differential routing, enhanced security, and a guarantee of interoperability with other networks. Similarly, services continue to adopt IP infrastructures. By doing so, these services benefit from open standards-based interfaces, session control functions and management, logical subscriber databases, and simplified interaction with other IP-based services and applications.

Two implementations that exemplify the concept of IP convergence are discussed in this part. Cell phone and public switched telephone networks traditionally have implemented individual, and often proprietary, infrastructures. However, as the need for reduced processing costs and increased scalability have arisen, telecommunication providers have begun converging these services and the networks across which the services are provided to the IP model, comprising a protocol called Voice over Internet Protocol (VoIP). Similarly, television broadcasting companies now need the capability of reaching an ever-increasing population. Additionally, some of this population reside in remote areas where the creation of an infrastructure to provide television is not feasible. However, by implementing television networks using an IP model, access can be provided to a much larger set of subscribers. Furthermore, wireless capabilities using IP can be much more cheaply implemented, providing access to remote areas. These concepts and more make up the protocol called IP Television (IPTV).

As networks and services continue to converge, more and more resources will become available through networks. Such an influx of networked data provides numerous new security exposures, inherently increasing the need for stronger, more flexible security models. Therefore, there have been many advances in user authentication, data privacy, and data integrity. This includes adapting existing technologies—and creating new ones—in the realms of not only cryptology and authentication algorithms, but also in network access control.

Finally, as users and organizations become more dependent on the continually growing number of networked resources, the importance of uninterrupted access to these resources becomes paramount. To accommodate this, advanced concepts of availability, scalability, and load balancing have arisen to ensure access to resources 24 hours a day and 7 days a week. These concepts not only include redundancy to account for hardware failure, but also embrace models of workload distribution in order to ensure proper utilization of these resources.

20

Voice over Internet Protocol

This chapter contains the following topics:

- An overview of VoIP
- Benefits of VoIP
- VoIP components
- SIP technology
- Processing flows
- Architecture and protocol standards
- Relevant RFCs

© Copyright IBM Corp. 1989-2006. All rights reserved.

20.1 Voice over IP (VoIP) introduction

As the number of services implemented over IP-based networks continues to grow, the inclusion of voice communication services in this progression is only natural. This convergence manifests itself through Voice over Internet Protocol (VoIP). In traditional implementations, communication services occur across the circuit protocols of the public switched telephone network (PSTN). Conversely, VoIP works by encoding voice information into a digital format, which can be carried across IP networks in discrete packets. Additional distinctions between VoIP and PSTN exist and we discuss these side-by-side.

By nature, analog voice signals are perceived as a sequence of sound vibrations. Because vibrations cannot be directly represented in a digital format, modern telecommunications must replicate voice signals by sampling human sound. These samples are then encoded in digits to replicate the sample's sounds. This digitized voice signal can then be carried by a transport network, using either a carrier network or a public internet.

Traditional telephone networks allocate dedicated circuits for voice communications. This occurs whenever someone picks up the phone and dials someone. Because the circuits are dedicated to the user, the quality of the voice signals can be guaranteed. However, for this same reason, the connection's potential bandwidth is wasted whenever there are periods of silence during the communication.

This wasted bandwidth does not occur when using VoIP, because packet networks allow sharing of transport facilities. However, the guarantee of service is lost as well. To counteract this, many carriers are in the process of transforming their network into IP packet-switching facilities to increase efficiency, while retaining control of the overall network resources. This ensures that transport resources will be available during a phone call.

Note: Sending voice in packet format through IP network is only one of many steps in making Voice over IP a possibility. There are additional functions necessary in setting up a call (signaling) when converting from an IP network to a telephone network (gateways).

20.1.1 Benefits and applications

The key drivers in VoIP are cost benefits, because cost savings are the primary short-term reason to converge voice, data, and video onto a single IP network (with everything over same infrastructure). Corporations have deployed

large-scale data networks, expecting cost savings and reduced expenses resulting from a merged multimedia network. Specifically:

- For the users, voice calls using the Internet are a means to reduce costs, avoiding tolls on the PSTN (toll bypass) and long-distance voice networks.
- For a pure VoIP service provider, minimal infrastructure ownership can be achieved by allowing the client to tap into the existing Internet service provider access and transport.
- For cable operators, VoIP enables the reuse of the existing infrastructure while rolling out new services.
- For network carriers, VoIP service is mainly a tool to counter the threat from new service providers while increasing the efficiency of their own infrastructure.

New applications provide improvements to existing telecom services. Key VoIP applications include:

- Telecom trunk replacements by using Ethernet trunk cards VoIP Private Branch Exchange (PBX) switches
- Toll bypass services for multinational corporations by linking offices from different cities or countries
- Rapid application deployment, made possible by following the IT and data networking world in combination with the use of open APIs and standards

Finally, applications and service features can be distributed throughout the network. Traditional central office switching equipment can be replaced by less expensive servers in handling the call processing and routing.

Archived

20.1.2 VoIP functional components

Figure 20-1 illustrates the main components of a VoIP networking system.

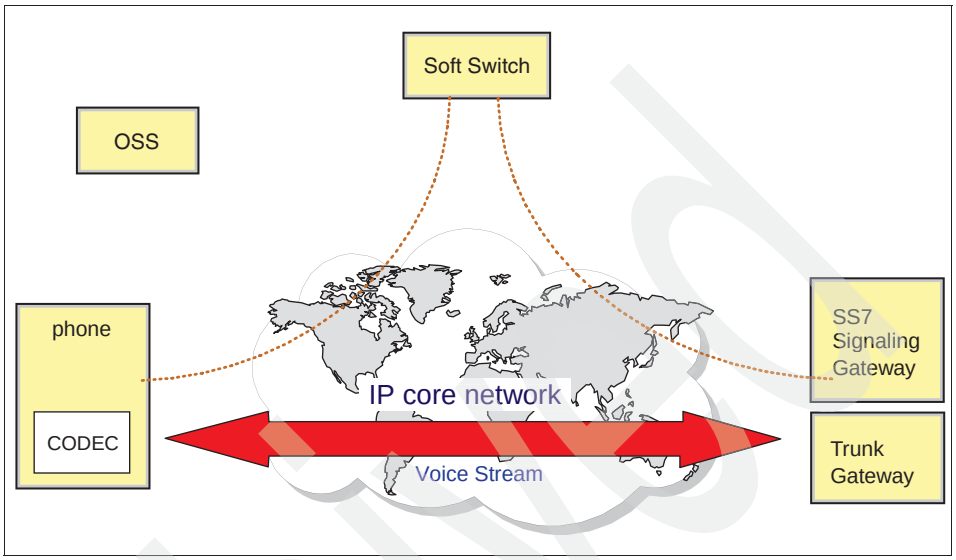


Figure 20-1 VoIP architecture

Softswitch

In the traditional PSTN model, a switchboard was used to connect two separate phone circuits into one contiguous line. In the VoIP model, this can be done through software. This concept is the softswitch, and it controls connections between the circuit and packet networks in segments where end-to-end VoIP is not yet available. The softswitch can be broken into two entities: the Call Agent and the Media Gateway.

Media Gateway

The Media Gateway is used to connect different physical networks in order to provide end-to-end connectivity. It functions very similarly to a typical network switch in that it can create a heterogeneous link between endpoints, regardless of the network media in between. In addition, the Media Gateway can also connect a VoIP circuit to a PSTN circuit, allowing the use of VoIP despite gaps in networks, or even when only one of the endpoints if VoIP enabled. This functionality is provided transparently, and therefore users have no need to understand or be aware of the topologies over which their communications are transferred.

Call Agent

The Call Agent serves several purposes. It can use a TCP/IP link to manage multiple Media Gateways. Primarily, however, it manages the following functions:

- Billing
- Call routing
- Network signalling

Phone handset

There are two different types of handsets:

Softphone The relationship between a traditional phone handset and a softphone is similar to the relationship between a switchboard and a softswitch. The functionality of the softphone is provided through software, often running on a personal computer. The hardware involved typically consists of any combination of computer sound cards, headphones, or speakers, and a microphone. This implementation is currently included in common instant message programs and is often used in PC-to-PC calls.

Hardphones The hardphone and the traditional handset correspond almost one-to-one. The hardphone is simply a traditional telephone handset that is VoIP capable.

Both types of VoIP phones are telephones capable of carrying voice traffic using the Session Initiation Protocol (see 20.2, “Session Initiation Protocol (SIP) technologies” on page 730) over a broadband connection.

Analog Terminal Adapters (ATA)

Analog Terminal Adapters (ATA) perform analog and digital conversions between a traditional analog phone and the broadband modem, allowing users to use the VoIP service with an existing phone. The ATA functions much as a Media Gateway does, translating data between analog and digital communication. ATAs typically have two ports: a telephone jack (FXS port) and a LAN port. This enables a user to plug into the FXS port an analog telephone, and then attach the ATA to the network. The ATA can then communicate with a VoIP server using protocols such as H.323, SIP, MGCP, or IAX.

Coder-decoder (CODEC)

The processing of digitizing a voice sample, or converting a digitized signal into an analog signal, requires the use of a coder-decoder. Each softphone, hardphone, or ATA must implement a CODEC in order to implement VoIP. There

is a set of international CODEC standards, any of which can be used to perform this translation. These standards include:

G.711 The G.711 (64 kps) standard, which encodes non-compressed speech streams running at 64 Kbps. It is widely used in the telecommunications field because it improves the signal-to-noise ratio without increasing the amount of data. It is required in the old Voice over IP network standards such as H.323.

G.726 G.726 is an ITU-T ADPCM speech CODEC standard covering the transmission of voice at rates of 16, 24, 32, and 40 kbps. It was introduced to supersede both G.721, which covered ADCPM at 32 kbps, and G.723, which described ADPCM for 24 and 40 kbps. G.726 also introduced a new 16 kbps rate. The four bit rates associated with G.726 are often referred to by the bit size of a sample, which are 2 bits, 3 bits, 4 bits, and 5 bits, respectively.

G.728 G.728 is a ITU-T standard for speech coding operating at 16 kbps. The technology used is LD-CELP, Low Delay Code Excited Linear Prediction. Delay of the CODEC is only five samples (0.625 ms). The linear prediction is calculated backwards with a 50th order LPC filter. The excitation is generated with gain scaled vector quantization (VQ). The standard was finished in 1992 in the form of algorithm exact floating point code. In 1994, a bit exact fixed point CODEC was released. G.728 passes low bit rate modem signals up to 2400 bps. Also, network signalling goes through. The complexity of the CODEC is 30 MIPS. Two kb of RAM is needed for codebooks.

G.723.1 G.723.1 is an audio CODEC for voice that compresses voice audio in chunks of 30 milliseconds. A look-ahead of 7.5 ms duration is also used. Music or tones such as DTMF or fax tones cannot be transported reliably with this CODEC, and thus some other method such as G.711 or out-of-band methods should be used to transport these signals. G.723.1 is mostly used in Voice over IP (VoIP) applications for its low bandwidth requirement. It became an ITU-T standard in 1995. The complexity of the algorithm is below 16 MIPS. 2.2 kilobytes of RAM is needed for codebooks.

There are two bit rates at which G.723.1 can operate:

- 6.3 kbps (using 24-byte chunks) using a MPC-MLQ algorithm (MOS 3.9)
- 5.3 kbps (using 20-byte chunks) using an ACELP algorithm (MOS 3.62)

G.723.1 is intended to standardize videoconferencing and is used in real-time video and teleconferencing applications where reduced bandwidth and very high quality voice is required. Therefore, this

technology is ideal for Internet video, audio, and telephony applications and will enable interoperability between telephony applications both on, and off, the net.

G.729 G.729 is an audio data compression algorithm for voice that compresses voice audio in chunks of 10 milliseconds. Music or tones such as DTMF or fax tones cannot be transported reliably with this CODEC, and thus uses G.711 or out-of-band methods to transport these signals.

G.729 is mostly used in Voice over IP (VoIP) applications for its low bandwidth requirement. Standard G.729 operates at 8 kbps, but there are extensions that provide 6.4 kbps and 11.8 kbps rates for marginally worse and better speech quality, respectively. G.729a, which is compatible with G.729, but requires less computation, is also very common. This lower complexity is not free because speech quality is marginally worsened.

G.729 (8 kps) provides near toll-quality performance. This standard was selected as the alternate default encoder for the IMTC VoIP Forum.

The G.729 CODEC is a high-complexity algorithm, and G.729a is a medium-complexity variant of G.729. The undistorted, toll-quality speech ensured by this standard makes it a popular choice for applications such as teleconferencing, visual telephony, VoIP, cellular phones, and other wireless applications where quality, delay, and bandwidth are important.

Core IP network

Given the name *core IP network*, it would be easy to assume that this refers simply to the Internet between two endpoints. While this might be true to some extent, the core IP network generally refers to a subset of the Internet that can guarantee a quality of service when transporting a stream of IP packets between voice applications. In many cases, this subset is proprietary and made available by a specific ISP offering voice services.

Signaling gateway

In order to implement VoIP communications, the functionality to notify an endpoint that another endpoint desires communication is necessary (such as by causing the recipient's phone to ring). This is called *signalling*. However, the method by which signalling is implemented in PSTN circuits differs from those used in VoIP circuits. As such, a gateway must be used to translate between the two in instances where there is no purely VoIP connection. This is the signalling gateway, which is capable of interworking standard signalling protocols such as CAS, DTMF, R1, R2, DTMF, ISDN, C5, and C7.

Operational support system

OSS provides user account setup, VoIP service provision, and fulfillment. If there are any issues in VoIP services, OSS provides trouble ticketing, troubleshooting, and service assurance. OSS also provides billing and other operational supports.

20.2 Session Initiation Protocol (SIP) technologies

Developed in 1996, the Session Initiation Protocol was originally designed as a means of notifying or inviting users to Internet multicast and broadcast sessions. This design, an early form of Internet signalling, made SIP very useful for signalling in VoIP, and thus most VoIP developers have adopted SIP as the core IP telephony standard. The most current RFC definition for SIP, published in 2002, is RFC 3261. However, it has been updated several times since 2002, as illustrated in Table 20-1.

Table 20-1 RFC updates to SIP

Update description	RFC	Year
SIP-Specific Event Notification	3265	2002
S/MIME Advanced Encryption Standard for SIP	3853	2004
Discussion of problems identified with SIP's Non-INVITE Transaction	4320	2006

SIP provides control over multimedia sessions. Implemented at the application layer, it is capable of establishing, modifying, and terminating sessions. In the context of IP telephony, these sessions are the VoIP "calls" themselves, and SIP is used to place the calls, modify them in-session (for example, inviting other users for features such as three-way calling), and to hang up. Also integrated into SIP's design is mobile capability, because SIP handles name mapping and redirection servers. This allows users to use IP telephony without regard to their physical or network location.

Primary SIP functions include:

User location and name translation
: User location and name translation is used to ensure that the call reaches the user.

User availability
: User availability determines whether the called party is willing to engage in communication.

Users capabilities Used for feature negotiation. This allows the group on the call to agree on the different features supported. If a certain CODEC rate is not supported by SIP, there is room for negotiation.

Session setup Session setup is used for establishing the communication session parameters at both ends.

Session management Session management is used for modifying session parameters and invoking services.

SIP applications are based on a client/server architecture (see 11.1.1, “The client/server model” on page 408), consisting of:

- The user agent (or proxy) as the client
- The network server

User agent or proxy

A user agent (SIP client) or proxy acts as the endpoint entity. Through request and response exchanges, the user agent can initiate and terminate sessions. In a SIP network, this functionality is found in devices such as:

- IP phones
- Telephony gateways
- Call agents
- Automated answering servers

A user agent resides in every SIP end station (phone or PC softphone). It contains two components:

- A user agent client (UAC) is responsible for issuing SIP requests.
- A user agent server (UAS) is responsible for receiving and responding to requests.

Network servers

SIP network servers are used to support advanced calling functions. There are multiple types of network servers:

- Redirect serverware provides a user agent placing a call with the address of a desired target, much like an application can determine a host’s IP address using the NSLOOKUP utility. The request is sent to a SIP server, which returns the necessary addresses, or zero if there is no known address.

Archived

- The proxy server provides application layer routing of SIP requests and responses. When the server receives a request, it forwards the request to a next hop server having more information about the location of the called device. These are also intermediary entities, which can act as both a client and a server in order to make requests on behalf of other clients. These requests are either handled internally or by passing them on to other servers.
- Registrar servers are used to record the SIP address and the associated IP address of a device.
- The application server provides call processing, personal and group services, and service management.
- The network server provides policy-based call routing and translations.
- The media server provides specialized media resources, interactive voice response, three-way conferencing, and other regulatory requirements.

SIP network servers usually implement a combination of the different server types.

20.2.1 SIP request and response

There are seven types (methods) of SIP requests:

INVITE	Indicates a user or service is being invited to participate in a call session.
ACK	Confirms that the client has received a final response to an INVITE request.
BYE	Terminates a call, and can be sent by either the caller or the callee.
CANCEL	Cancels any pending searches but does not terminate a call that has already been accepted.
OPTIONS	Queries the capabilities of servers.
PRACK	Provisional acknowledgement.
REGISTER	Registers the address listed in the “To” header field with a SIP server. Registrations can require authentication.

The following types of SIP responses are used.

SIP 1xx	Informational responses (for example, 180 Ringing)
SIP 2xx	Successful responses (for example, 200 OK)
SIP 3xx	Redirection responses (for example, 302 Temporarily Moved)

Archived

SIP 4xx	Client failure responses (for example, 404 User Not Found)
SIP 5xx	Server failure responses
SIP 6xx	Global failure responses

20.2.2 Sample SIP message flow

An invitation occurs when one SIP endpoint (user A) “invites” another SIP endpoint (user B) to join in a call:

1. During this process, a user sends an INVITE message to the AS requesting that user B join/setup a call.
2. The server processes the request and returns an appropriate response (for example, 100 Trying or 487 User Busy).

An example of SIP basic call flow is illustrated in Figure 20-2 on page 734:

1. A SIP device sends the server an invite (INVITE100 Trying).
2. The server sends back a response (180 Ringing or 183 Session Progressing).
3. A PRACK 200 OK (optional) can be sent, which is followed by an answer 200 OK (INVITE Answer) with an acknowledgement.
4. At this point, a two-way voice path is established for Realtime Transport Protocol (RTP) stream.
5. When a user terminates the call with a BYE, it is acknowledged with an ACK.

Archived

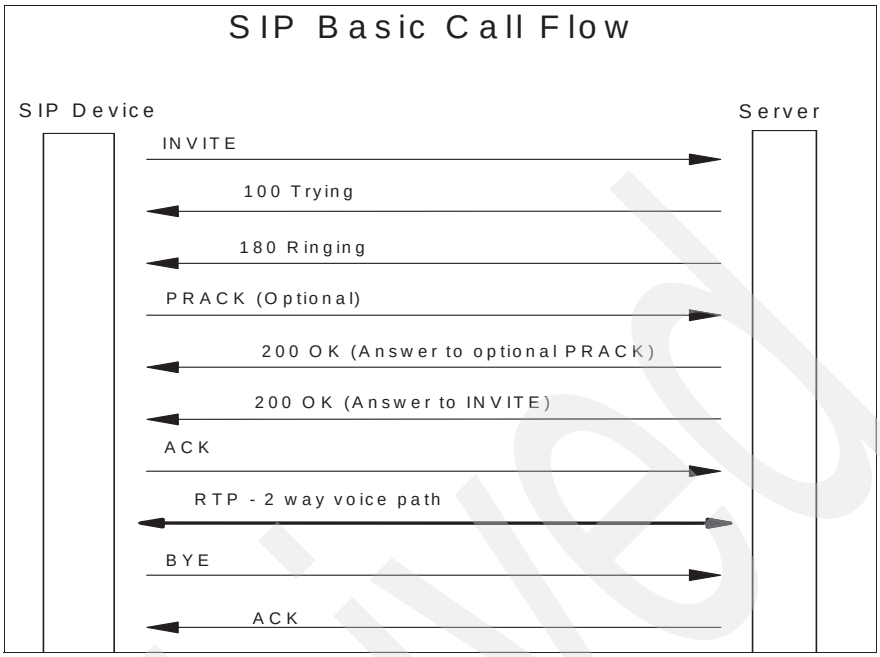


Figure 20-2 An example of SIP request/response flows

20.2.3 SIP protocol architecture

SIP architects used the Internet model during the development of the standard. As such, the architecture of SIP is similar to the architecture of HTTP, and both systems use a *request-response* model for communication. The calling device sends a request to the called device. The called device then chooses to accept or reject the request and returns the response to the initiator.

Another feature of HTTP leveraged by SIP is the HTTP-based message formats. SIP reuses most of the header fields, encoding rules, and error codes of HTTP, providing a readable, text-based format for displaying information.

A SIP address is referred to as a SIP Uniform Resource Locator (SIP-URL). It has the format sip:username@hostname.domain. This address is used to identify an individual device or group of devices. It enables a user to initiate a call by selecting a link in a standard browser.

SIP protocol stack

Figure 20-3 depicts the overall architecture for the SIP-based protocol stack.

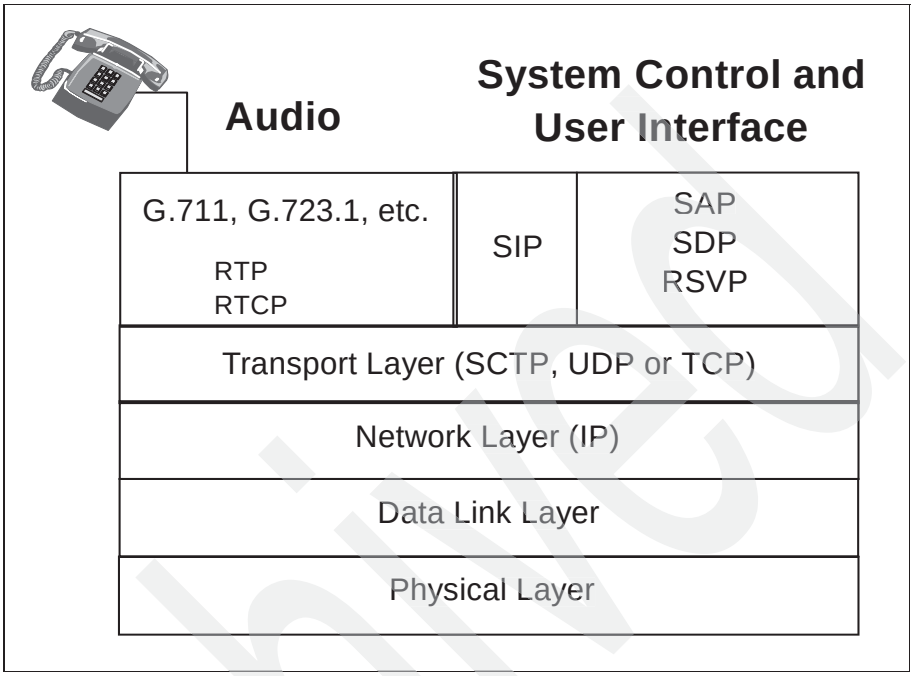


Figure 20-3 SIP protocol stack

In a SIP environment, voice calls are carried using RTP. SIP relies on additional standards to provide signaling and control functions. These include:

- The G.7xx series of protocols provide basic CODEC functions, as described in 20.1.2, “VoIP functional components” on page 726.
- The Real-time Transport Protocol (RTP) allows the receiver to detect packet loss. It also provides timing information that allows the receiver to compensate for variable packet arrival times (that is, jitter). Details of RTP are in RFC 3550.
- The Real-time Control Protocol (RTCP) accompanies RTP, and is also defined in RFC 3550. It obtains the network status and QoS feedback information. RTCP provides the function to manage real-time delivery of packets.

- Stream Control Transmission Protocol (SCTP) provides a transport mechanism between SIP entities that exchange a large number of messages. Because SIP is transport-independent, support of SCTP is a relatively straightforward process, nearly identical to support for TCP. SCTP specifications are in RFC 3286.
- Resource Reservation Protocol (RSVP) is defined in RFC 2205, and is used to create quality of service. The request is processed by each node along the session path and devices reserve the appropriate resources to support the application data stream.
- Session Description Protocol (SDP) is a protocol that is defined in RFC 2327. It provides session announcements and session invitations in a multimedia environment. This enables the recipients of the session announcement to participate in the session.
- Session Announcement Protocol (SAP) is another protocol for advertising multicast conferences and sessions and is defined by RFC 2974. It announces the existence of long-lived, wide-area, multicast sessions.

20.3 Media Gateway Control Protocol (MGCP)

MGCP is the Internet Engineering Task Force's standard for multimedia conferencing over IP. Development of the architectures that form MGCP started in mid-1998, but the Internet Engineering Task Force (IETF) published the latest MGCP standard, RFC 3435, in 2003.

Unlike the SIP architecture, in which the intelligence resides in the endpoints, the MGCP model assumes that the core contains the faculties for implementing enhanced services. As a result, MGCP endpoint phones have minimal technical features. This aspect is beneficial for carriers because it enables the delivery of enhanced services through low-cost endpoints.

MGCP provides the capabilities to:

1. Determine the location of the target endpoint.
2. Determine the media capabilities of the target endpoint through the Session Description Protocol (SDP).
3. Determine the availability of the target endpoint.
4. Establish a session between the originating and target endpoint.

20.3.1 MGCP architecture

An alternate signalling protocol, MGCP is an ASCII-based application layer control protocol that employs both master/subordinate relationships and client/server relationships. The master/subordinate relationships exist between the call agent (the master component) and multiple media gateways (subordinate components), and is used in setting up calls to and from endpoints. The components in this relationship then comprise the server entity in the client/server relationships. Endpoints acting as clients invoke the services of the call agent, and subsequently the media gateways, in order to establish, maintain, and terminate calls with another endpoint.

Call agents

The intelligence required to perform call operations resides in the call agent, and therefore, the call agent directs the operations of a gateway. These devices are also responsible for signaling and call processing functions.

An MGCP network does not need to integrate with an H.323 network. However, because the MGCP call agent implements signaling functions, it can appear as an H.323 gatekeeper (see 20.5, “ITU-T recommendation H.323” on page 739).

Media gateway

A media gateway is responsible for converting between the audio signals used in the PSTN and the data packets used in IP networks. Examples of gateways include:

- Trunking gateways provide an interface between a traditional PSTN telephone network and a VoIP network. These gateways typically manage a large number of circuits.
- Residential gateways provide an interface between a traditional telephony user (analog RJ-11) and a VoIP network. These include interfaces to cable modems and xDSL devices.

Because MGCP is a master/subordinate protocol, a gateway is expected to execute the commands provided by a call agent.

20.3.2 MGCP primitives

MGCP messages are composed from a short list of primitives:

RQNT “NotificationRequest” instructs the MGCP device to watch for specific events.

NTFY “Notify” informs the call agent when requested events occur.

CRCX “CreateConnection” creates a connection to an endpoint.

MDCX “ModifyConnection” changes the parameters associated with an established connection.

DLCX “DeleteConnection” deletes an existing connection. ACK returns call statistics.

AUEP “AuditEndpoint” audits an existing endpoint.

AUCX “AuditConnection” audits an existing connection.

RSIP “RestartInProgress” is a gateway notification to the call agent that an media gateway or an endpoint is restarting or stopping.

Among those primitives, of specific interest are the notification messages. The MGCP device uses these messages to tell the call agent of a change of state and typically involve signaling or events. We describe examples of both:

- Signals include ringing, distinctive ringing, ringbacktone, dial tone, intercept tone, network congestion tone, busy tone, confirm tone, answer tone, call waiting tone, off-hook warning tone, pre-emption tone, continuity tone, continuity test, and dial tones.
- Events include the on-hook transition, off-hook transition, flash hook, and receipt of dial digits.

20.4 Media Gateway Controller (Megaco)

Similar to the use of MGCP in an IP telephony network, Megaco addresses the relationship between a gateway and a call agent. However, unlike MGCP, Megaco supports a broader range of networks, including ATM. This makes the protocol applicable to a much wider range of media gateways.

After the MGCP standard was approved, additional proposals and enhancements were submitted to the IETF. The Megaco protocol combines these different standards into a single architecture and is the result of joint development by the ITU and the IETF.

20.4.1 Megaco architecture

Megaco has been architected to support gateways ranging in size from a single port to thousands of ports.There are two components in a standard Megaco network:

- Terminations
- Contexts

Terminations

Terminations represent streams entering or leaving the gateway. Some terminations, typically representing ports on the gateway, are automatically created when the device activates. Other terminations are created when required. These represent transient flows through the network, such as RTP traffic flows and VoIP streams.

Contexts

Terminations are placed into contexts. This occurs when two or more termination streams are connected together. A simple call might have two terminations per context (one termination representing the end device and the other termination representing the connection to the network). A conference call can have numerous terminations per context, each termination representing one leg of the conference.

Megaco uses a series of commands to manipulate terminations and contexts. For example, the ADD command adds a termination to a context, and the MOVE command moves a termination from one context to another. These commands are sent from the call agent to the gateway.

20.5 ITU-T recommendation H.323

H.323 is an ITU-T standard originally developed for multimedia videoconferencing over packet-switched networks and was later extended to include IP telephony requirements. At the time of this writing, it is the most widely deployed call processing protocol.

The initial version of H.323 was developed very rapidly. Work started on ITU-T in May 1995 and was completed by June of 1996. Since that time, three additional versions have been published.

20.5.1 H.323 architecture

There are four components in a standard H.323 network:

- Terminals
- Gateways
- Gatekeepers
- Multipoint control units

Archive

Terminals

Terminals are the LAN client endpoints in an IP telephony network. An H.323 terminal can communicate with other H.323 terminals, an H.323 gateway, or an H.323 multipoint control unit (MCU).

All terminals must support voice communication, though support for video and data is optional. H.323 specifies the operations required for different types of terminals to work together, and all H.323 terminals must support the set of protocols described in 20.5.2, “H.323 protocol stack” on page 741.

Additionally, H.323 terminals support multipoint conversations and provide the ability to initiate ad-hoc conferences. They also have multicast features that allow multiple people to participate in a call without centralized mixing or switching.

Gateways

Gateways enable standard telephones to use VoIP services. They provide communication between H.323 terminals and terminals connected to either an IP-based network or another H.323 gateway. The gateway functions as a translator, providing the interface between the PSTN and the IP network.

The gateway is responsible for mapping the call signaling and control protocols between dissimilar networks. It is also responsible for media mapping (for example, multiplexing, rate matching, audio transcoding) between the networks.

Gatekeepers

Gatekeepers are the most important component in an H.323 environment. A network of H.323 terminals and gateways under the control of a particular gatekeeper forms an integrated subnetwork within the larger IP network environment. The gatekeeper's functions include:

- Directory server: Using information obtained during terminal registration, this function translates an H.323 alias address to an IP (transport) address. This enables the user to have meaningful, unchanging names to reference other users in the system. These names are arbitrary and can appear similar to those used in e-mail or voice mail applications.
- Supervisory: The gatekeeper can grant permission to make a call. This can be used to apply bandwidth limits, reducing the likelihood of congestion within the network.
- Call signaling: The gatekeeper can perform call routing functions to provide supplementary services. It can also provide multipoint controller functionality, supporting calls with a large number of participants.
- Call management: The gatekeeper can perform call accounting and call management.

Multipoint control units (MCUs)

A multipoint control unit (MCU) is an endpoint in the network. It provides the ability for three or more terminals or gateways to participate in a multipoint conference. An MCU provides conference management, media switching, and multipoint conferencing.

There are three types of multipoint conferences:

- Centralized: All terminals have point-to-point communication streams with the MCU. The MCU is responsible for management of the conference. It receives, processes, and sends the voice packets to other terminals.
- Decentralized: Terminals communicate directly with each other. An MCU is not directly involved.
- Mixed multipoint: This represents a mixture of centralized and decentralized conferences. The MCU ensures operations of the conference are transparent to the each terminal.

20.5.2 H.323 protocol stack

The H.323 specification is an umbrella recommendation. Various components of H.323 are defined by other ITU-T standards. Figure 20-4 on page 742 shows the suite of H.323 definitions:

- H.225 call control: This protocol provides call signaling and control functions. In networks without a gatekeeper, call signaling messages are passed directly between the calling and called endpoints. In networks that contain a gatekeeper, these messages are initially exchanged between the calling endpoint and the gatekeeper. H.225 call control messages use TCP transport services.
- H.225 registration, admission, and signaling (RAS) control: The RAS channel is used for communication between the endpoints and the gatekeeper. The protocol defines standard procedures for gatekeeper discovery, endpoint registration, endpoint location, and admissions control. H.225 RAS messages use UDP transport services.
- H.245 conference control: This specifies the protocol used after a call establishment phase has completed. H.245 negotiates the media channels used by RTP and RTCP. The protocol defines standard procedures for exchanging capabilities, determining master and subordinate assignments, and controlling conferences.
- The other protocols are already described in 20.2.3, “SIP protocol architecture” on page 734.

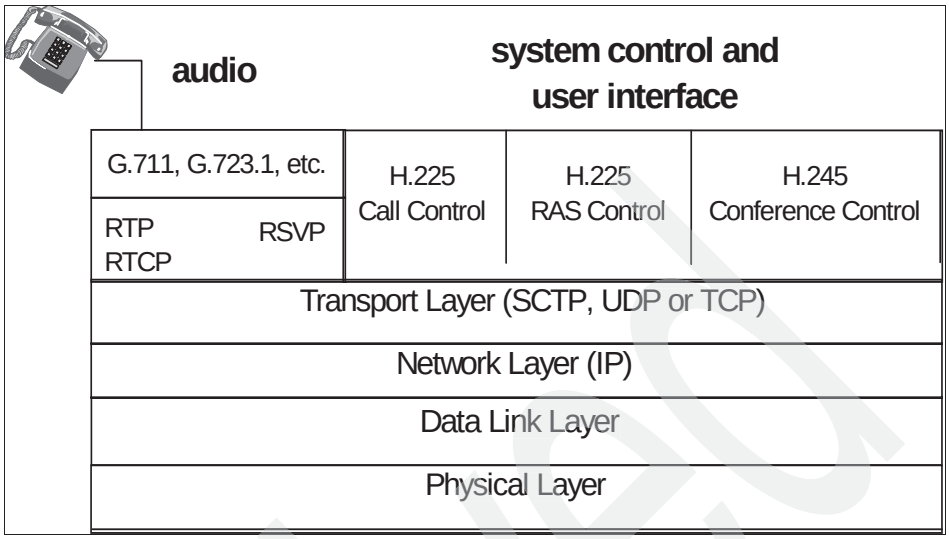


Figure 20-4 H.323 protocol stack

20.6 Summary of VoIP protocols

An IP telephony environment consists of a number of individual protocols. These protocols interoperate in a hierarchical fashion to provide the required services. The protocol interaction can be represented as a stack, similar to the method used to represent many other communications systems.

One way to depict the stack is to divide the protocols into two functional sections, which are illustrated in Figure 20-5 on page 743. The two sections of the protocol stack provide the following functions:

- Transport and quality protocols

  These protocols are used to transport the voice traffic.

- Signaling and support protocols

  The majority of IP telephony activities have been devoted to developing signaling protocols.

Archived

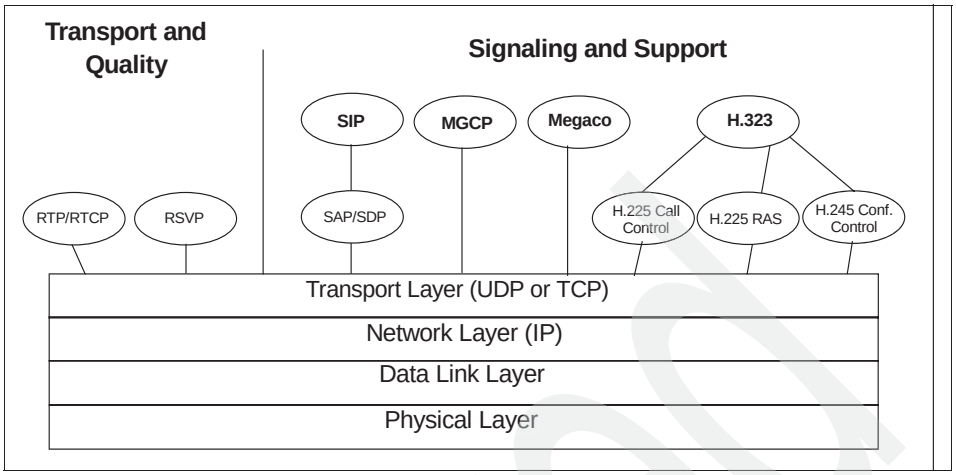


Figure 20-5 The functional sections of the IP stack

The signaling and control protocols locate users, negotiate capabilities, and manage voice calls. There are currently four major signaling protocols:

- Session Initiation Protocol (SIP)
- Media Gateway Control Protocol (MGCP)
- ITU-T Recommendation H.248/Media Gateway Controller (Megaco)
- ITU-T) Recommendation H.323

These protocols are incompatible and often provide redundant functions. When they are used in the same environment, communication between these protocols requires a converter or gateway.

20.7 RFCs relevant to this chapter

The following RFCs provide detailed information about Voice over Internet Protocols presented throughout this chapter:

- RFC 3261 – SIP: Session Initiation Protocol (June 2002)
- RFC 4168 – The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP) (October 2005)
- RFC 3264 – An Offer/Answer Model with Session Description Protocol (SDP) (June 2002)
- RFC 3265 – Session Initiation Protocol (SIP)-Specific Event Notification (June 2002)

- RFC 3266 – Support for IPv6 in Session Description Protocol (SDP) (June 2002)
- RFC 3331 – Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) - User Adaptation Layer (September 2002)
- RFC 3332 – Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA) (September 2002)
- RFC 3435 – Media Gateway Control Protocol (MGCP) Version 1.0 (January 2003)
- RFC 3436 – Transport Layer Security over Stream Control Transmission Protocol (December 2002)
- RFC 3482 – Number Portability in the Global Switched Telephone Network (GSTN): An Overview (February 2003)
- RFC 3525 – Gateway Control Protocol Version 1 (June 2003)
- RFC 3550 – RTP: A Transport Protocol for Real-Time Applications (July 2003)
- RFC 3690 – IP Telephony Requirements for Emergency Telecommunication Service (ETS) (February 2004)
- RFC 3976 – Interworking SIP and Intelligent Network (IN) Applications (January 2005)
- RFC 4083 – Input 3rd-Generation Partnership Project (3GPP) Release 5 Requirements on the Session Initiation Protocol (SIP) (May 2005)
- RFC 4123 – Session Initiation Protocol (SIP)-H.323 Interworking Requirements (July 2005)
- RFC 4168 – The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP) (October 2005)
- RFC 4396 – RTP Payload Format for 3rd Generation Partnership Project (3GPP) Timed Text (February 2006)

Archived

21

Internet Protocol Television

This chapter discusses the following topics:

- What is IPTV?
- Business benefits and applications
- IPTV requirements on infrastructure
- V^2oIP and IPTV components
- Architecture, protocols, and technologies
- IPTV standards

© Copyright IBM Corp. 1989-2006. All rights reserved.

21.1 IPTV overview

IPTV is an IP network application that supports several high quality video and audio standards (for example, MPEG and H.261) for live video, scheduled video, and video on demand.

Generally speaking, IPTV refers to the distribution of television and video signals in parallel with the Internet connection supplied by a broadband operator using the same IP (and mostly the same VoIP) packet network infrastructure.

Figure 21-1 illustrates IPTV application architecture.

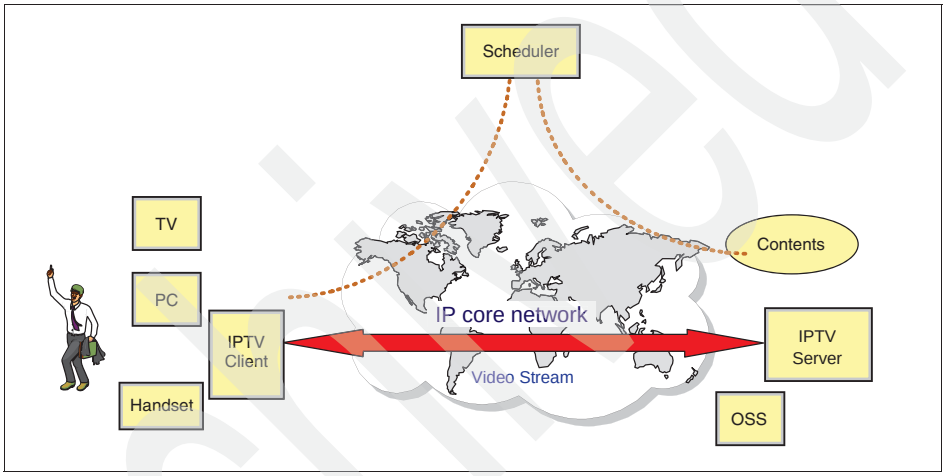


Figure 21-1 IPTV application architecture

As shown in the IPTV application architecture diagram in Figure 21-1, the following activities can occur:

- The users can receive and view the contents on a home TV, a PC, or even with a hand-held device on the move. The devices are loaded with an IPTV client.
- Content servers house the title collections and the feeds from content producers. The TV contents are usually encoded in a standard format.
- The contents can be broadcast or on demand distribution, all controlled by the schedule server.
- The IPTV server handles user requests and the flows are chopped into a stream of IP packets.

- The IP packets flow through a core IP network. Unlike a voice stream that only requires 64 kbps or less with compression, a compressed video stream needs over a megabit per second to satisfy users' TV experience. This is the reason for a broadband network connection.
- To ensure smooth delivery of the TV service, a service provider might need to provide dedicated connections and resources to ensure no interruption in the user experience. Therefore, a guaranteed path from the content source to the final distribution might have to be under watch by a network service provider.
- Complex Operation Support Systems (OSS) are set up to fulfill service ordering, service assurance, and billing functions.

21.1.1 IPTV requirements

There are a number of requirements for IPTV to be successful:

- Quality of service or QoS

  Some specific QoS mechanisms are required in a bandwidth-constrained access. A bandwidth allocation supported by a session-based admission control function is a reasonable assumption.

  A flow has to be set up along the whole transmission path. The quality requirements can be adjusted from time to time.

  Note that such a dynamic QoS scheme might require changes in the QoS policy to be propagated throughout the network infrastructure.

- Multicast

  Use of multicast is a convenience for the operator to minimize network resources and not an essential component of the service definition. Protocol Independent Multicast (PIM) can help decouple the underlying routing protocol from broadcast services, using dense mode or sparse mode through the planning of IPTV coverage and usage in the network infrastructure.

  The IPTV service needs to be available whether the content is unicast (as in video on demand) or multicast (as in TV broadcast).

- Authentication/authorization/identity

  The IPTV service must be authorized for a specific facility. More sophisticated individualized authorization models might be required for some variants of the service, for example, parental controls.

Archived

- Communication context

  This is similar to the presence service in VoIP convergence. The Internet TV stream can be one element within a communication context. Consider two teenagers watching the same movie (in their respective houses) and instant messaging each other at significant moments during the show. An alternative example is the TV stream as a picture-in-picture option within a multimedia conferencing service.

- Nomadicity

  For an Internet TV service, nomadicity normally is interpreted as the user being able to access the same service profile from a different location. Performance-to-cost ratio changes are likely to result in handheld devices or portable terminals capable of participating in Internet TV sessions.

- Emergency services

  IPTV needs to continue existing television services to provide content overlays notifying of emergency conditions (for example, a thunderstorm warning).

- Content and service discovery

  Assisted by a centralized or distributed directory, the user is able to search and locate the video content from the terminals. The search can be text-, picture-, or video-based. The content and the service available based on the content is reported back to the user as the result of search.

  As the cable and phone service providers are moving into the content service, they face integration of various content providers. For example, a new search for a TV title results in multiple IP flows to various servers, some of which might contain the same content with varying format, abbreviation, or technology.

- Digital rights management

  IP network security and application security support DRM technology that allows certification of the legitimacy of the original source of the video content, authentication and authorization of the user to view the content, and clear accounting among content creators, producers, traders, and users.

- Session management

  Standard session management control commands must be built on top of the SIP or RTCP or other protocols in order for users to initiate and control a content delivery. Examples include play, rewind, forward, pause, stop, fast rewind, and fast forward.

To meet the IPTV service requirements, a set of enabling capabilities have to provide middleware in the SP space. Those enabling technologies are directly supported by a number of IETF protocols including Real-Time Protocol,

Real-Time Control Protocol, and Stream Control Transmission Protocol. We explain protocols and technologies in later sections of this chapter.

21.1.2 Business benefits and applications

As we mentioned earlier in this chapter, IPTV offerings stem from competing needs between cable and phone service providers. Of course, new IPTV technologies and applications, in turn, benefit content providers, as well as new user experience.

IP services provide a number of benefits:

- To the users, IPTV technology enables them to select the content they want and pick the time they want to watch. They can use the devices they want and available to them. Thin clients will be embedded in various devices.
- The content providers add new distribution opportunities. They can provide targeted custom TV services and commerce with the collection of usage data.
- The application service providers can simplify integration of IP content and services (such as IP packetized weather, games, photos, and music on TV). For example, TV can be integrated with broadband home devices though a set-top box (STB). The IP STB is the TV's gateway to digital music, home videos, and photos.
- The network infrastructure gets filled with new traffic to increase utilization of the ever-growing fiber bandwidth.
- The cable and phone service providers find a level play field to compete in trip-play for data, voice, and video.
- Traditionally, cable and wireless offered separate (no overlap) services with different network infrastructures. As they move toward the same service space and with a common IP domain, we (the authors) predict integration and mergers across cable and phone industry, as a consequence of the same triple-play ground place.

Archive

21.2 Functional components

Figure 21-2 illustrates the major functional components in IPTV.

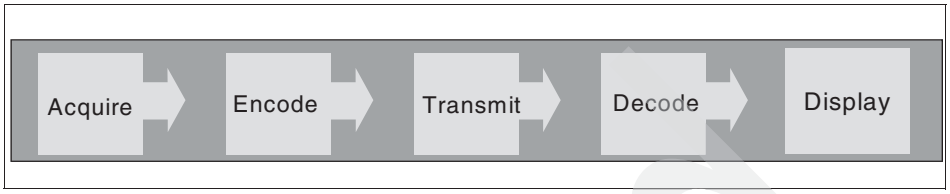


Figure 21-2 IPTV functional components

21.2.1 Content acquisition

The first step in IPTV is acquisition of the contents to be broadcast. Here we focus on the IP networking technical aspects:

- Multiple IP network interconnections are required between the content service provider infrastructure and that of the IPTV application service provider.
- IP video differs from a traditional cable or satellite TV architecture in that the radio frequency or RF signal is sampled, packetized, and encapsulated into packets for delivery over a IP-based broadband network.
- The acquired contents have to be in one or more video content warehouses, where the titles have to be searchable. For this purpose, traditional DNS can be extended to some aspects of the service level.
- Normally, the distribution network is arranged as a hierarchical layout. The top distribution router has to handle a higher volume of IP traffic, and the regional hub can store and route to local areas. Transport between the top level and the regional hubs can be dedicated IP networks, while the lower hierarchical distribution layer can use a shared IP network infrastructure.

21.2.2 CODEC (encode and decode)

A CODEC is the combination of coder and decoder. The CODEC can also compress and expand the video signals.

There are several reasons for encoding the signals:

- Compression

  The bandwidth required to transmit uncompressed video exceeds the limitations of most network technologies today.

- Encryption

  Content providers want to protect their assets from unauthorized use. At some point or another, every signal transmission is either compressed or decompressed using Moving Pictures Expert Group (MPEG) standards, the latest one providing built-in Digital Right Management capabilities.

In IPTV applications, MPEG is a very efficient way of compressing video and audio signals to a level at which digital transmission becomes possible. Here we summarize the MPEG standards:

- MPEG-1 was the first release and was designed for bit rates up to 1.5 Mbps.
- MPEG-2 mainly improved on MPEG-1 for video encoding. Depending on the settings, the bit rate can be more than 20 Mbps, but is more likely to be around 4 Mbps for standard content.
- MPEG-3 is not used. Because MPEG-2 proved so successful, the definitions for HDTV, which were destined to become MPEG-3, were absorbed in MPEG-2. Because there were minimal changes to the audio encoding, it is common to use MPEG-2 video and MPEG-1 audio.
- MPEG-4 is designed for a wide range of bit rates and treats the video component as being made up of objects rather than pixels. For additional details about MPEG4, see 21.3.6, “Moving Picture Experts Group (MPEG) standards” on page 767.

21.2.3 Display devices and control gateway

In the IPTV era, any IP devices can be used to receive the IP stream and play back the contents. This imposes access connectivity requirements for the IPTV transport network interfaces:

- Wireline Ethernet has to be supported.
- Wireless, for example, 3G/4G wireless, interfaces have to be supported.
- Wireless LAN (IEEE 802.16) has to be available.

The devices are loaded with an “IPTV client” to provide user registration, content scheduling, and interaction with IPTV servers for presentation control. Presentation control directly impacts the user viewing experience.

Typically, in a home environment, a set-top box or a dedicated desk server is dedicated as a gateway for dispatch and control configurations.

Service providers can provision the services by invoking the gateway device. New OSS platforms have to be designed and developed for IPTV service provisioning and troubleshooting.

21.2.4 IP (TV) transport

IPTV requires large bandwidth (5 Mbps to 30 Mbps). Table 21-1 breaks down the bandwidth requirements for a hypothetical scenario.

Table 21-1 Sample bandwidth requirements

Application	Bandwidth
One (1) high-definition TV (HDTV)	10 Mbps
Two (2) other TVs	2 x 1.5 Mbps = 3 Mbps
(Subtotal) Three (3) TVs in an average US family	**13 Mbps**
Recording for later viewing	**1.5 Mbps**
High speed internet browsing	**3 Mbps**
Total bandwidth	13+1.5+3 = **17.5 Mbps**

Network access upgrades to fiber are necessary. There are two types of broadband network upgrades:

- Fiber to the home (FTTH)
- Fiber to the neighborhood (FTTN)

The core IP network must provide video quality of service (QoS) precedence class marking. The traffic type marking is according to QoS standards, such as the Diffserv model described in Chapter 8, “Quality of service” on page 287.

21.3 IPTV technologies

There are several key technologies necessary for a successful IPTV implementation:

- The transport technologies including RTP, RTCP, SCTP, described later in this chapter.
- Encoding technology, MPEC, described later in this chapter.
- Session description technology (SDP), described in this chapter.
- Multicast setup and control technology (IGMP), described in 3.3, “Internet Group Management Protocol (IGMP)” on page 119. Protocol Independent Multicast (PIM) is also used in ITPTV to improve distribution efficiency. PIM is described in 6.7, “Protocol Independent Multicast (PIM)” on page 261.

- QoS technologies including RSVP, IntServ, DiffServ, discussed in Chapter 8, “Quality of service” on page 287.

21.3.1 Summary of protocol standards

Figure 21-3 shows a summary of the IPTV protocol stack.

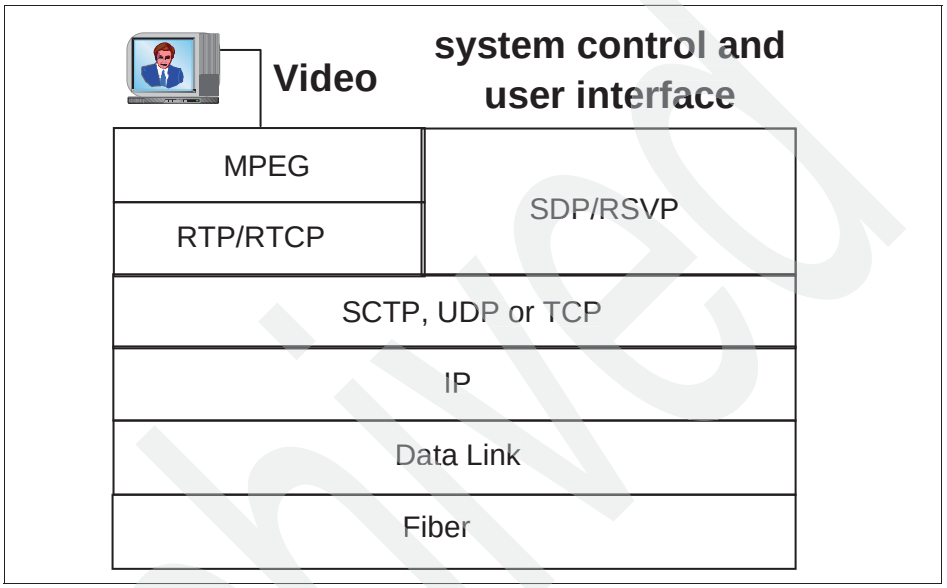


Figure 21-3 IPTV protocol stack

21.3.2 Stream Control Transmission Protocol

For multimedia over IP applications, Stream Control Transmission Protocol (SCTP) is a new transport protocol at the same layer as TCP and UDP. SCTP provides functions for association management, sequence delivery, message chunk building, packet validation, and path management.

SCTP has been designed with improved features for real-time services, including:

- Fast retransmit

  SCTP can quickly determine the loss of a packet, because of its usage of selective acknowledgement (SACK) and a mechanism that sends SACK messages faster than normal when losses are detected.

- Congestion control

  SCTP maintains congestion control over the entire association. Congestion state (measured in terms of the UDP retransmit interval) is computed on a transaction-by-transaction basis, rather than across all transactions. Therefore, congestion control performance is similar to opening N parallel TCP connections, as opposed to sending N messages over one TCP connection.

- Transport-layer fragmentation

  SCTP provides transport-layer fragmentation as TCP does and unlike UDP where fragmentation occurs at the IP layer. IP fragmentation increases the likelihood of having packet losses and makes NAT and firewall traversal difficult, if not impossible.

- Non-blocking head of the line

  SCTP is message-based, as opposed to TCP, which is stream-based. This allows SCTP to separate different signalling messages at the transport layer. SCTP can deliver signalling messages to the application as soon as they arrive (when using the unordered service). The loss of a signalling message does not affect the delivery of the rest of the messages.

- Easier parsing

  SCTP allows easier parsing of messages at the application layer. There is no need to establish boundaries (typically using Content-Length headers) between different messages.

- Multihoming

  An SCTP connection can be associated with multiple IP addresses on the same host. Data is always sent over one of the addresses, but if it becomes unreachable, data sent to one can migrate to a different address. This improves fault tolerance; network failures making one interface of the server unavailable do not prevent the service from continuing to operate.

For additional details about SCTP, refer to RFC 2960.

21.3.3 Session Description Protocol

Session Description Protocol (SDP) provides a standard representation to convey media details, transport addresses, and other session description metadata to the participants.

SDP is purely a format for session description; it does not incorporate a transport protocol. SDP is intended for describing multimedia sessions for the purposes of

session announcement, session invitation, and other forms of multimedia session initiation. For example:

- RTSP client and server negotiate an appropriate set of parameters for media delivery, using SDP syntax to describe those parameters.
- At in SIP session initiation, messages carry session descriptions that allow participants to agree on a set of compatible media types. These session descriptions are commonly formatted using SDP.
- In a multicast session announcement, a distributed session directory can communicate the relevant session setup information to prospective participants. An instance of such a session directory periodically sends packets containing a description of the session to a well-known multicast group. SDP provides the recommended session description format for such session announcements.

An SDP session description includes:

- Session name and purpose
- Times the session is active (start, stop, recurrence)
- The media comprising the session (MPEG video over RTP)
- Information needed to receive those media (addresses, ports, formats, and so on)
- Information about the bandwidth to be used by the session
- Contact information for the person responsible for the session

An SDP session description is entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attribute names use only the US-ASCII subset of UTF-8, but textual fields and attribute values can use the full ISO 10646 character set.

An SDP session description consists of a number of lines of text of the form:

```
<type>=<value>
```

Where <type> *must* be exactly one case-significant character. Certain types are required. The optional items are marked with an asterisk "*" in the following list (Table 21-2).

Table 21-2 SDP session description

Type	Description
v	Protocol version
o	Originator and session identifier

Type	Description
s	Session name
i*	Session information
u*	URI of description
e*	E-mail address
p*	Phone number
c*	Connection information, not required if included in all media
b*	Zero or more bandwidth information lines
t	Time the session is active
r*	Zero or more repeat times
z*	Time zone adjustments
k*	Encryption key
a*	Zero or more session attribute lines
m	Media name and transport address
i*	Media title
c*	Connection information, optional if included at session level
b*	Zero or more bandwidth information lines
k*	Encryption key
a*	Zero or more media attribute lines

For additional details, refer to RFC 4566.

21.3.4 Real-Time Transport Protocol (RTP)

The Real-Time Transport Protocol (RTP) provides the transport of real-time data packets. To accommodate new real-time applications, the architecture was intentionally left incomplete. Unlike conventional protocols, RTP is tailored through modifications and additions to headers as needed. This allows the protocol to easily adapt to new audio and video standards.

RTP implements the transport features needed to provide synchronization of multimedia data streams.

Consider an application using both video and audio components. RTP can mark the packets associated with the individual video and audio streams. This allows the streams to be synchronized at the receiving host. Figure 21-4 shows the operation of RTP in a multimedia transmission. Audio and video data are encapsulated in RTP packets prior to transmission from the sender to the receiver.

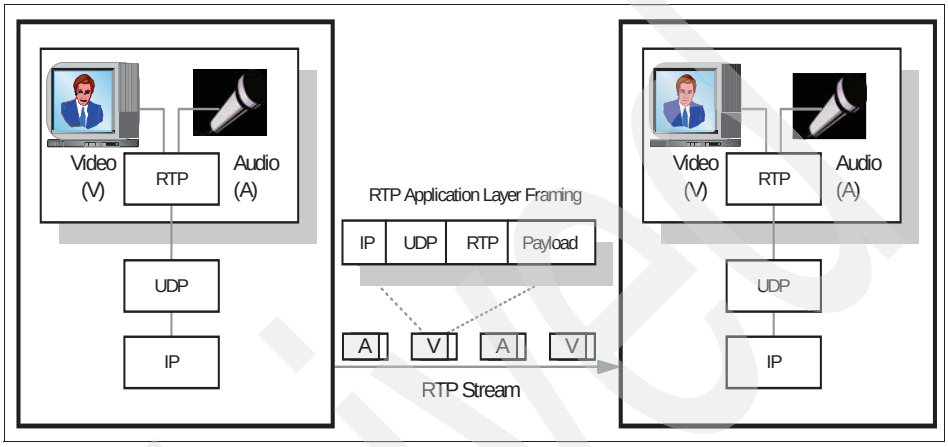


Figure 21-4 RTP operation

If the multimedia application does not use RTP services, the receiver might not be able to associate the corresponding audio and video packets. This can be attributed to the varying levels of network performance provided during a multimedia session. Congestion or other transient conditions within the environment can cause packets to be lost or reordered during transit. This can delay delivery of packets by varying amounts of time. This behavior causes quality problems with typical multimedia applications.

The standards specify that RTP can be used with any appropriate network or transport protocol. In practice, multimedia applications typically use RTP in conjunction with UDP. This allows the application to use the multiplexing and checksum services provided by UDP. RTP is often implemented to support multicast applications.

The RTP protocol alone does not include any mechanism to provide guaranteed delivery or other quality of service functions. The standard does not prevent out-of-sequence packet delivery, nor does it assume that the underlying network is reliable and delivers packets in-sequence. RTP also does not prevent the occurrence of network congestion. Designers of each application must determine if these levels of service are acceptable.

RTP header format

The header of an RTP packet has the following format, illustrated in Figure 21-5.

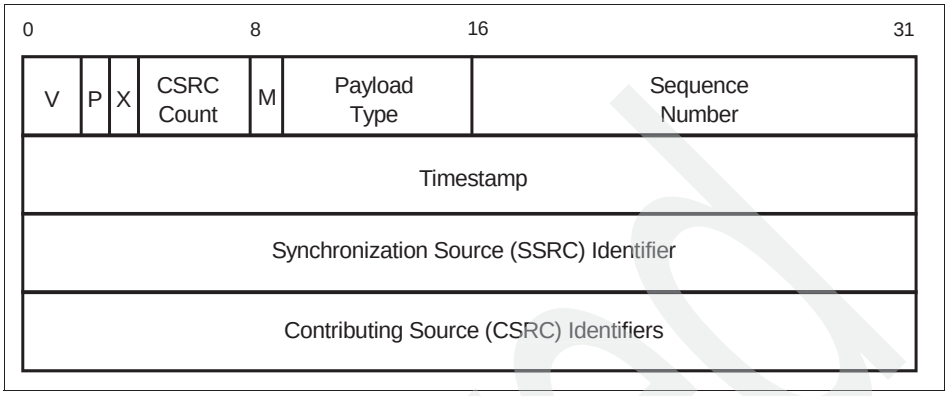


Figure 21-5 RTP header format

The first 12 octets are required in every RTP packet. The list of CSRC identifiers is present only when inserted by a mixer. The individual fields have the following interpretation:

V Indicates the RTP version.

P Contains the padding bit. If this field is set, the packet contains a set of padding octets that are not part of the payload. This function is used by certain encryption algorithms.

X Contains the extension bit. If this field is set, a header extension follows the fixed header.

CSRC count This field contains the number of contributing source identifiers that follow the fixed header.

M This field allows significant events to be marked in the packet stream (that is, frame boundaries).

Payload type Specifies the format of the payload in the RTP packet. An RTP sender emits a single RTP payload type at any given time. Refer to “Payload type identification” on page 759 for further information regarding the contents of the payload type field.

Sequence number The sequence number is used by the receiver to restore packet sequence and detect packet loss. Refer to “Sequence numbering” on page 761 for further information regarding the contents of the sequence number field.

Timestamp The time stamp contains a value representing the time when the payload data was sampled. Refer to “Time stamps” on page 761 for further information regarding the timestamp field.

SSRC identifier The synchronization source is a randomly chosen identifier for an RTP host. All packets from the same source contain the same SSRC identifier. Each device in the same RTP session must have a unique SSRC identifier. This enables the receiver to group packets for playback.

CSRC identifiers The contributing source field contains a list of the sources for the payload in the current packet. This field is used when a mixer combines different streams of packets. The information contained in this field allows the receiver to identify the original senders.

Protocol services

RTP provides end-to-end transport services for applications transmitting real-time data. These services include:

- Payload type identification
- Sequence numbering
- Time stamps

Payload type identification

An RTP packet can contain portions of either audio or video data streams. To differentiate between these streams, the sending application includes a payload type identifier within the RTP header. The identifier indicates the specific encoding scheme used to create the payload. The receiving application uses this identifier to determine the appropriate decoding algorithm. Table 21-3 shows the payload types that are supported.

Table 21-3 RTP payload type

Audio		Video	
Payload type	**Encoding name**	**Payload type**	**Encoding name**
0	PCMU	24	unassigned

Archived

Audio		Video	
1	1016	25	CelB
2	G726-32	26	JPEG
3	GSM	27	unassigned
4	G723	28	nv
5	DVI4 (8 KHz)	29	unassigned
6	DVI4 (16 KHz)	30	unassigned
7	LPC	31	H261
8	PCMA	32	MPV
9	G722	33	MP2T
10	L16 Stereo	34	H263
11	L16 Mono	35-71	unassigned
12	QCELP	72-76	reserved
13	reserved	77-95	unassigned
14	MPA	96-127	dynamic
15	G728	dynamic	BT656
16	DVI4	dynamic	H263-1998
17	DVI4	dynamic	MP1S
18	G729	dynamic	MP2P
19	reserved	dynamic	BMPEG
20-23	unassigned		
dynamic	GSM-EFR		
dynamic	L8		
dynamic	RED		
dynamic	VDVI		

This list is maintained by the Internet Assigned Numbers Authority (IANA). It may be enhanced as new audio and video formats are developed. Although RTP was primarily designed to support multimedia data, it is not limited to audio and video traffic. Any application producing continuous data streams can use RTP services.

Several encoding schemes can specify a dynamic payload type. Systems deploying these encoding schemes dynamically agree on the payload type identifier.

Sequence numbering

Sequence numbers are used by the receiving RTP host to restore the original packet order. The receiver is able to detect packet loss using the information in this field.

The sequence number increments by one for each RTP data packet sent. The initial value of the sequence number is randomly determined. This makes hacking attacks on encryption more difficult. A random number is used even if the source device does not encrypt the RTP packet. The packets can flow through a translator that does provide encryption services.

Time stamps

Time stamps are used in RTP to synchronize packets from different sources. The time stamp represents the sampling (creation) time of the first octet in the RTP data packet. It is derived from a clock that increments monotonically and linearly. The resolution of the timer depends on the desired synchronization accuracy required by the application.

It is possible that several consecutive RTP packets have the same time stamp. For example, this can occur when a single video frame is transmitted in multiple RTP packets. Because the payloads of these packets were logically generated at the same instant, the time stamps remain constant. The initial value of the time stamp is random. Figure 21-6 on page 762 shows an example of time stamp generation in a video application.

Archived

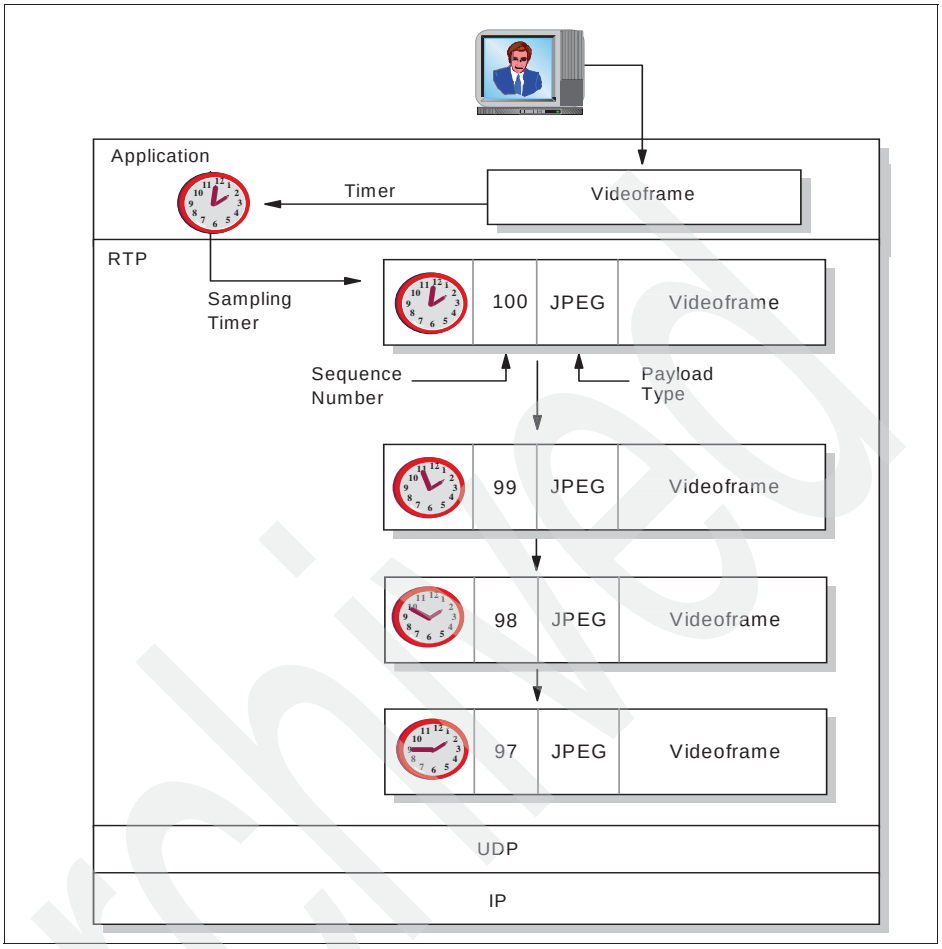


Figure 21-6 RTP packet generation in a video application

21.3.5 Real-Time Control Protocol

The Real-Time Control Protocol (RTCP) monitors the quality of service provided to existing RTP sessions.

The primary function of RTCP is to provide feedback about the quality of the RTP data distribution. This is comparable to the flow and congestion control functions provided by other transport protocols. Feedback provided by each receiver is used to diagnose distribution faults. By sending feedback to all participants in a session, the device observing problems can determine if the problem is local or remote. This also enables a managing entity (that is, a network service provider) that is not a participant in the session to receive the feedback information. The

network provider can then act as a third-party monitor to diagnose network problems.

RTCP uses a UDP connection for communication. This is separate from any UDP connection used by the RTP protocol. Figure 21-7 shows the cooperation of the RTP and the RTCP protocols.

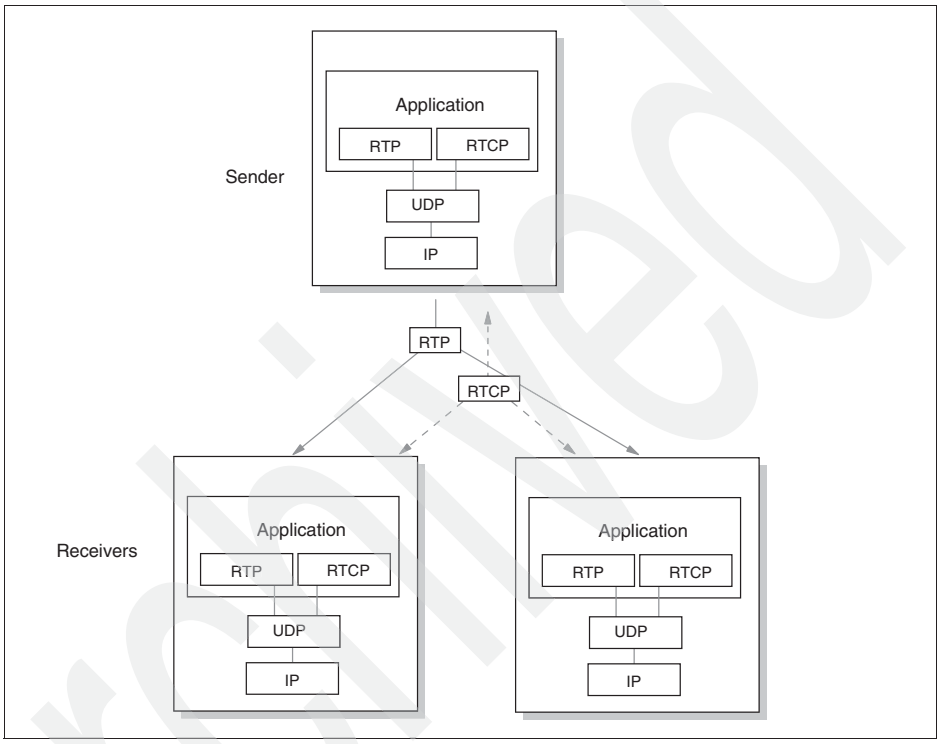


Figure 21-7 RTCP and RTP packet delivery

The RTCP architecture defines five types of control information used to report current performance:

Sender report An RTCP sender report is sent by the source of an RTP data stream. It provides the transmission and reception statistics observed by the sender. This report is sent as a multicast packet processed by all RTP session participants. We describe the packet format for this report in “RTCP packet format” on page 764.

Archived

Receiver report An RTCP receiver report provides reception statistics for participants that are not active senders. A sender report is issued if a device has sent any data packets during the interval since issuing the last report; otherwise, a receiver report is issued.

Source description report A source description packet is used by an RTP sender to provide local capability information. The currently defined source descriptions include:

- **CNAME** A unique name for the source
- **NAME** The real name of the source
- **EMAIL** The e-mail address of the application user
- **PHONE** The phone number of the application user
- **LOC** The geographic location of the application user
- **TOOL** The specific application or tool name
- **NOTE** Additional notes about the source
- **PRIV** Private extensions

BYE The RTCP bye message is used by a source when it leaves a conference. This is used when a mixer shuts down. The BYE message is used to indicate all sources contributing to the session.

APP The APP packet is intended for experimental use as new applications and features are developed. After testing and if wider use is justified, it is recommended that each APP packet be registered with the Internet Assigned Numbers Authority.

RTCP packet format

Figure 21-8 on page 765 shows the format of a RTCP sender report. The report is grouped in three sections. The first section contains the header. This section specifies the packet type, length, and sender identification.

The second section contains sender information. The third section contains receiver report blocks. The packet can contain several receiver report blocks. This enables the sender to report feedback from RTP packets received from other senders.

The format of the sender report is shown for clarity. The formats for the other four report types are similar.

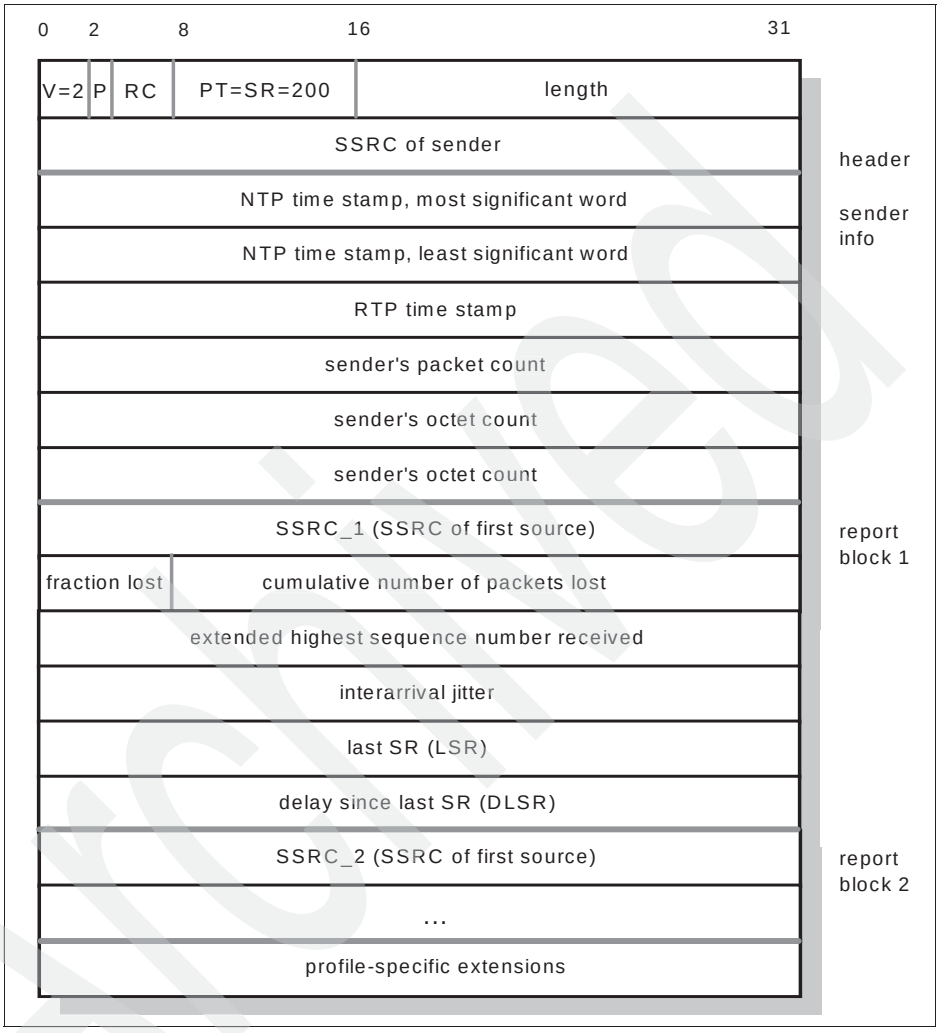


Figure 21-8 RTCP packet format

The fields in the packet are used to:

V	Indicates the version.
P	Indicates if additional padding is located at the end of the packet.
RC	Contains the number of report blocks in this packet.
PT	Contains the report type.

Length Contains the packet length.

SSRC of sender Contains the SSRC identifiers of the host sending this packet.

NTP timestamp Contains the absolute time reported by NTP. This protocol counts the number seconds since January 1, 1900.

RTP timestamp Contains the time stamp from the RTP packets according to the sender.

Sender's packet count

Contains the total number of RTP data packets transmitted by the sender since the start of the transmission.

Sender's octet count

Contains the total number of payload bytes transmitted by the sender since the start of the transmission.

SSRC_n Contains the SSRC identifier of another RTP sender from which this sender has received packets. The number of report blocks with different sender SSRCs depends on the number of other sources that were heard by this sender since the last report.

Fraction lost Contains the fraction of RTP data packets that were lost since the previous sender report or receiver report was sent from the source SSRC_n.

Cumulative number of packets lost

Indicates the total number of lost RTP packets from source SSRC_n.

Extended highest sequence number received

Contains the highest sequence number that was received in an RTP packet from the source SSRC_n.

Interarrival jitter Contains the estimated variance of the interarrival times from the appropriate source. If the packets arrive regularly, the jitter value is zero. If the packets arrive irregularly, the jitter value is high.

Last SR timestamp (LSR)

Contains the middle 32 bits from the 64-bit NTP time stamp received in the last RTCP sender report packet from the source SSRC_n.

Delay since last SR (DLSR)

Contains the delay between receiving the last SR packet from the source SSRC_n and the sending of the current exception report block in units of 1/65536 seconds.

21.3.6 Moving Picture Experts Group (MPEG) standards

MPEG is a standard for encoding and decoding video in a compressed format. The latest version is MPEG-4.

It was jointly developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) as the product of a collective partnership effort known as the Joint Video Team (JVT). ISO/IEC MPPEG-4 and ITU-T H.264 are technically identical. The following list describes the standards:

- MPEG-1: Initially designed as a video and audio compression standard. Later, it became the standard for video CD. This standard included the MPEG Audio Layer 3 (MP3) compression format.
- MPEG-2: Designed as the TV compression format for transport of video and audio for broadcast-quality television. Used for over-the-air digital television ATSC, DVB and ISDB, digital satellite TV services, digital cable television signals, and (with slight modifications) for digital video disk (DVD).
- MPEG-3: Originally designed for HDTV. It was obsoleted because MPEG-2 was sufficient for HDTV.
- MPEG-4: The latest version of video compression with low-bit rate encoding and security support. Additional higher efficiency video standards are included for Advanced Simple Profile (ASP) and Advanced Video Coding (AVC). AVC works well on a very wide variety of networks and systems (for example, for broadcast, DVD storage, RTP/IP packet networks, and ITU-T multimedia systems).

MPEC provides multipicture motion compensation using previously-encoded pictures as references in a much more flexible way than in past standards, allowing up to 32 reference pictures to be used in some cases (unlike in prior standards, where the limit was typically one or two). In certain types of scenes, for example, scenes with rapid repetitive flashing or back-and-forth scene cuts or uncovered background areas, it allows a very significant reduction in bit rate.

It uses variable block-size motion compensation (VBSMC) with block sizes as large as 16 by16 and as small as 4 by 4, enabling very precise segmentation of moving regions.

It allows weighted prediction, enabling an encoder to specify the use of a scaling and offset when performing motion compensation, and providing a significant benefit in performance in special cases such as fade-to-black, fade-in, and cross-fade transitions.

It uses data partitioning (DP), a feature providing the ability to separate more important and less important syntax elements into different packets of data, enabling the application of unequal error protection (UEP).

It employs a feature that serves to keep the ordering of the pictures and the values of samples in the decoded pictures isolated from timing information (allowing timing information to be carried and controlled/changed separately by a system without affecting decoded picture content).

Profiles

The standard includes the profiles for various specific classes of applications:

Baseline Profile (BP)	Primarily for lower-cost applications demanding less computing resources, this profile is used widely in video conferences and mobile applications.
Main Profile (MP)	Originally intended as the mainstream consumer profile for broadcast and storage applications, the importance of this profile faded when the High Profile was developed for those applications.
Extended Profile (XP)	Intended as the streaming video profile, this profile has relatively high compression capability and some extra tricks for robustness to data losses and server stream switching.
High Profile (HiP)	The primary profile for broadcast and disc storage applications, particularly for high-definition television applications (this is the profile adopted into HD DVD, for example).
High 10 Profile (Hi10P)	Going beyond today's mainstream consumer product capabilities, this profile builds on top of the High Profile by adding support for up to 10 bits per sample of decoded picture precision.
High 4:2:2 Profile (Hi422P)	Primarily targeting professional applications that use interlaced video, this profile builds on top of the High 10 Profile, adding support for the 4:2:2 chroma sampling format while using up to 10 bits per sample of decoded picture precision.

21.3.7 H.261

Due to its historical importance and its influence to MPEG development, we also want to mention the H.261 video decoding standard, which was the first practical digital video coding standard. It has since influenced all subsequent international video coding standards. MPEG-1, MPEG-2/H.262, H.263, and even H.264 have been based closely on its design.

The H.261 standard only specifies how to decode the video. Encoder designers were left free to design their own encoding algorithms, as long as their output was constrained properly to allow it to be decoded by any decoder made according to the standard. Encoders are also left free to perform any pre-processing they want to their input video, and decoders are allowed to perform any post-processing they want to their decoded video prior to display.

The coding algorithm uses a hybrid of motion compensated inter-picture prediction and spatial transform coding with scalar quantization, zigzag scanning and entropy coding:

- The inter-picture prediction removes temporal redundancy, with motion vectors used to help the CODEC compensate for motion.
- Transform coding using an 8x8 discrete cosine transform (DCT) removes the spatial redundancy.
- Scalar quantization is then applied to round the transform coefficients to the appropriate precision, and the quantized transform coefficients are zigzag scanned and entropy coded to remove statistical redundancy.

The standard supports CIF and QCIF video frames with luma resolutions of 352x288 and 176x144, respectively (and 4:2:0 sampling with chroma resolutions of 176x144 and 88x72, respectively). It also allows sending still picture graphics with 704x576 luma resolution.

The basic processing unit of the design is called a macroblock. Each macroblock consists of a 16x16 array of luma samples and two corresponding 8x8 arrays of chroma samples using 4:2:0 sampling and a <Yellow,Blue,Red> color space.

The refinements introduced in later standardization efforts have resulted in significant improvements in compression capability relative to the H.261 design. This has resulted in H.261 becoming essentially obsolete, although it is still used as a backward-compatibility mode in some video conferences systems and for some types of Internet video.

21.4 RFCs relevant to this chapter

The following RFCs provide detailed information about Internet Protocol Television as presented throughout this chapter:

- RFC 4566 – Session Description Protocol (July 2006)
- RFC 3550 – RTP and RTCP: A Transport Protocol for Real-Time Applications (July 2003)
- RFC 2960 – The Stream Control Transmission Protocol (SCTP) (October 2000)
- RFC 2368 – The Realtime Stream Protocol (RTSP) (April 1998)

Archived

22

TCP/IP security

This chapter discusses security issues regarding TCP/IP networks and provides an overview of solutions to prevent security exposures or problems before they occur. The field of network security in general and of TCP/IP security in particular is too wide to be dealt within an all encompassing way in this book, so the focus of this chapter is on the most common security exposures and measures to counteract them. Because many, if not all, security solutions are based on cryptographic algorithms, we also provide a brief overview of this topic for the better understanding of concepts presented throughout this chapter.

© Copyright IBM Corp. 1989-2006. All rights reserved.

22.1 Security exposures and solutions

This section gives an overview of some of the most common attacks on computer security, and it presents viable solutions to those exposures and lists actual implementations thereof.

22.1.1 Common attacks against security

For thousands of years, people have been guarding the gates to where they store their treasures and assets. Failure to do so usually resulted in being robbed, victimized by society, or even killed. Though things are usually not as dramatic anymore, they can still become very bad. Modern day IT managers have realized that it is equally important to protect their communications networks against intruders and saboteurs from both inside and outside. One does not have to be overly paranoid to find some good reasons as to why this is the case:

- Packet sniffing: To gain access to cleartext network data and passwords
- Impersonation: To gain unauthorized access to data or to create unauthorized e-mails by impersonating an authorized entity
- Denial-of-service: To render network resources non-functional
- Replay of messages: To gain access to information and change it in transit
- Password cracking: To gain access to information and services that would normally be denied (dictionary attack)
- Guessing of keys: To gain access to encrypted data and passwords (brute-force attack)
- Viruses: To destroy data
- Port scanning: To discover potential available attack points

Though these attacks are not exclusively specific to TCP/IP networks, they must be considered potential threats to anyone who is going to base their network on TCP/IP, which is the most prevalent protocol used. TCP/IP is an open protocol, and therefore, hackers find easy prey by exploiting vulnerabilities using the previous methods.

22.1.2 Solutions to network security problems

Network owners need to try to protect themselves with the same zealousness that intruders use to search for a way to get into the network. To that end, we provide some solutions to effectively defend a network from an attack, specifically against the attacks mentioned earlier. It has to be noted that any of

these solutions only solve a single (or a very limited number) of security problems. Therefore, consider a combination of several such solutions to guarantee a certain level of safety and security. These solutions include:

- Encryption: To protect data and passwords
- Authentication by digital signatures and certificates: To verify who is sending data over the network
- Authorization: To prevent improper access
- Integrity checking and message authentication codes: To protect against improper alteration of messages
- Non-repudiation: To make sure that an action cannot be denied by the person who performed it
- One-time passwords and two-way random number handshakes: To mutually authenticate parties of a conversation
- Frequent key refresh, strong keys, and prevention of deriving future keys: To protect against breaking of keys (cryptanalysis)
- Address concealment: To protect against denial-of-service attacks
- Disable unnecessary services: To minimize the number of attack points

Table 22-1 matches common problems and security exposures to the previous solutions.

Table 22-1 Security exposures and protections

Problem/exposure	Remedy
How to prevent a packet sniffer from reading messages?	Encrypt messages, typically using a shared secret key (secret keys offer a tremendous performance advantage over public/private keys).
How to distribute the keys in a secure way?	Use a different encryption technique, typically public/private key.
How to prevent keys from becoming stale, and how to protect against guessing of future keys by cracking current keys?	Refresh keys frequently and do not derive new keys from old ones (use perfect forward secrecy).
How to prevent retransmission of messages by an impostor (replay attack)?	Use sequence numbers (time stamps are usually unreliable for security purposes).
How to ensure that a message has not been altered in transit?	Use message digests (hash or one-way functions).
How to ensure that the message digest has not also been compromised?	Use digital signatures by encrypting the message digest with a secret or private key (origin authentication, non-repudiation).

Problem/exposure	Remedy
How to ensure that the message and signature originated from the desired partner?	Use two-way handshakes involving encrypted random numbers (mutual authentication).
How to ensure that handshakes are exchanged with the right partners (man-in-the-middle attack)?	Use digital certificates (binding of public keys to permanent identities).
How to prevent improper use of services by otherwise properly authenticated users?	Use a multilayer access control model.
How to protect against viruses?	Restrict access to outside resources; run anti-virus software on every server and workstation that has contact to outside data, and update that software frequently.
How to protect against unwanted or malicious messages (denial of service attacks)?	Restrict access to internal network using filters, firewalls, proxies, packet authentication, conceal internal address and name structure, and so on.
How to minimize the number of attack points?	Close all unnecessary services. Use encryption and encapsulation to run many services over a smaller number of ports.

In general, keep your network tight toward the outside, but also keep a watchful eye on the inside because most attacks are mounted from inside a corporate network.

22.1.3 Implementations of security solutions

The following protocols and systems are commonly used to provide various degrees of security services in a computer network. They are discussed at length throughout the rest of this chapter.

- IP filtering
- Network Address Translation (NAT)
- IP Security Architecture (IPSec)
- SOCKS
- Secure Shell (SSH)
- Secure Sockets Layer (SSL)
- Application proxies
- Firewalls
- Kerberos and other authentication systems (AAA servers)
- Secure Electronic Transactions (SET)

Figure 22-1 illustrates where these security solutions fit within the TCP/IP layers.

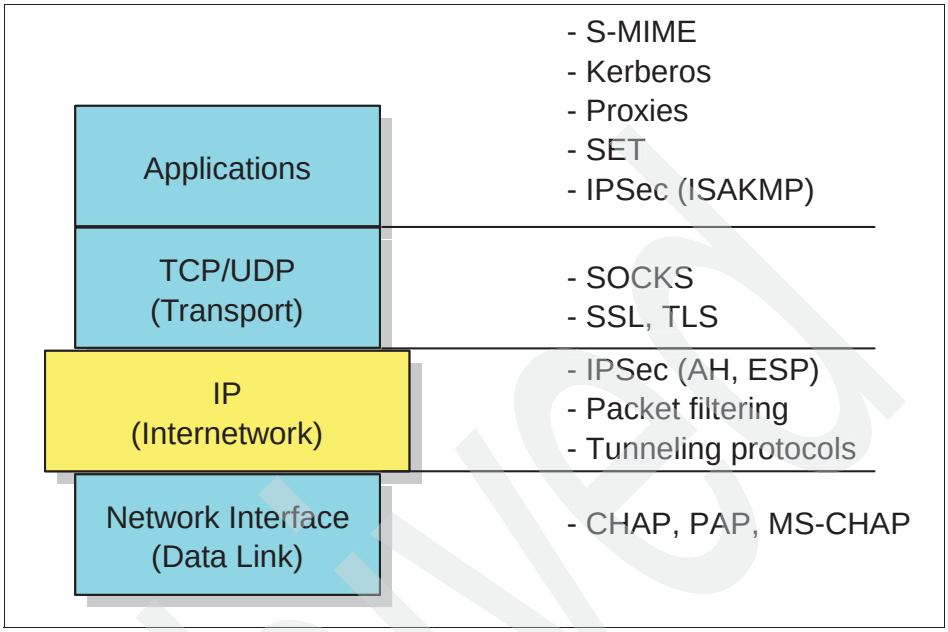


Figure 22-1 Security solutions in the TCP/IP layers

Table 22-2 summarizes the characteristics of some of the security solutions mentioned earlier and compares them to each other. This should help anyone who needs to devise a security strategy to determine what combination of solutions achieves a desired level of protection.

Table 22-2 Security solution implementations: A comparison

	Access control	Encryption	Authen-tication	Integrity checking	Perfect forward security	Address conceal-ment	Session monitoring
IP filtering	Y	N	N	N	N	N	N
NAT	Y	N	N	N	N	Y	Y (connection)
IPSec	Y	Y (packet)	Y (packet)	Y (packet)	Y	Y	N
SOCKS	Y	N	Y (client/ user)	N	N	Y	Y (connection)
SSL	Y	Y (data)	Y (system/ user)	y		n	y

	Access control	Encryption	Authen-tication	Integrity checking	Perfect forward security	Address conceal-ment	Session monitoring
Application proxy	Y	Normally no	Y (user)	Y	Normally no	Y	Y (connection and data)
AAA servers	y (user)	N	Y (user)	N	N	N	N

An overall security solution can, in most cases, only be provided by a combination of the listed options. Your particular security requirements need to be specified in a security policy and should be, for example, enforced by using firewalls and validated by using security health checking tools and vulnerability scanners.

22.1.4 Network security policy

An organization's overall security policy must be determined according to security and business needs analysis and based on security best practices. Because a firewall relates to network security only, a firewall has little value unless the overall security policy is properly defined.

A network security policy defines those services that will be explicitly allowed or denied, how these services will be used, and the exceptions to these rules. Every rule in the network security policy should be implemented on a firewall, remote access server (RAS), or both. Generally, a firewall uses one of the following methods.

Everything not specifically permitted is denied

This approach blocks all traffic between two networks except for those services and applications that are permitted. Therefore, each desired service and application is implemented one by one. No service or application that might be a potential hole on the firewall is permitted. This is the most secure method, denying services and applications unless explicitly allowed by the administrator. However, from the point of users, it might be more restrictive and less convenient.

Everything not specifically denied is permitted

This approach allows all traffic between two networks except for those services and applications that are denied. Therefore, each untrusted or potentially harmful service or application is denied one by one. Although this is a flexible and convenient method for the users, it can potentially cause some serious security problems, especially as new applications are introduced into the environment.

Remote access servers should provide authentication of users and should ideally also provide for limiting certain users to certain systems and networks within the corporate intranet (authorization). Remote access servers must also determine if a user is considered roaming (can connect from multiple remote locations) or stationary (can connect only from a single remote location), and if the server should use callback for particular users after they are properly authenticated.

Generally, anonymous access should at best, be granted to servers in a demilitarized zone (DMZ, see “Screened subnet firewall (demilitarized zone)” on page 808). All services within a corporate intranet should require at least password authentication and appropriate access control. Direct access from the outside should always be authenticated and accounted.

22.2 A short introduction to cryptography

The purpose of this chapter is to introduce the terminology and give a brief overview of the major cryptographic concepts that relate to TCP/IP security implementations. The information presented here only scratches the surface. Some issues are left open or not mentioned at all.

22.2.1 Terminology

Let us start with defining some very basic concepts.

Cryptography

Put simply, *cryptography* is the science of altering the appearance of data in an effort to keep data and data communications secure. To achieve this goal, techniques such as *encryption*, *decryption*, and *authentication* are used. With the recent advances in this field, the frontiers of cryptography have become blurred. Every procedure consisting of transforming data based on methods that are difficult to reverse can be considered cryptography. The key factor to strong cryptography is the difficulty of reverse engineering. You might be amazed to know that simple methods, such as password-scrambled word processor documents or compressed archives, can be broken in a matter of minutes by a hacker using an ordinary PC. *Strong* cryptography means that the computational effort needed to retrieve your cleartext messages without knowing the proper keys makes the retrieval infeasible. In this context, infeasible means something like this: If all the computers in the world were assigned to the problem, they would have to work tens of thousands of years until the solution was found. The process of retrieval is called *cryptanalysis*. An attempted cryptanalysis is an *attack*.

Encryption and decryption: Cryptographic algorithms

Encryption is the transformation of a cleartext message into an unreadable form in order to hide its meaning. The opposite transformation, which retrieves the original cleartext, is the *decryption*. The mathematical function used for encryption and decryption is the *cryptographic algorithm* or *cipher*.

The security of a cipher might be based entirely on keeping its functionality a secret, in which case it is a *restricted cipher*. There are many drawbacks to restricted ciphers. It is very difficult to keep an algorithm a secret when it is used by many people. If it is incorporated in a commercial product, it is only a matter of time and money before it is reverse engineered. For these reasons, the currently used algorithms are *keyed*, that is, the encryption and decryption makes use of a parameter, known as the *key*. The key can be chosen from a set of possible values, called the *keyspace*. The keyspace usually is huge, the bigger the better. The security of these algorithms rely entirely on the key, not on their internal secrets. In fact, the algorithms themselves are usually public and are extensively analyzed for possible weaknesses. The principle of keyed ciphers is shown in Figure 22-2.

Note: Do not trust new, unknown, or unpublished algorithms.

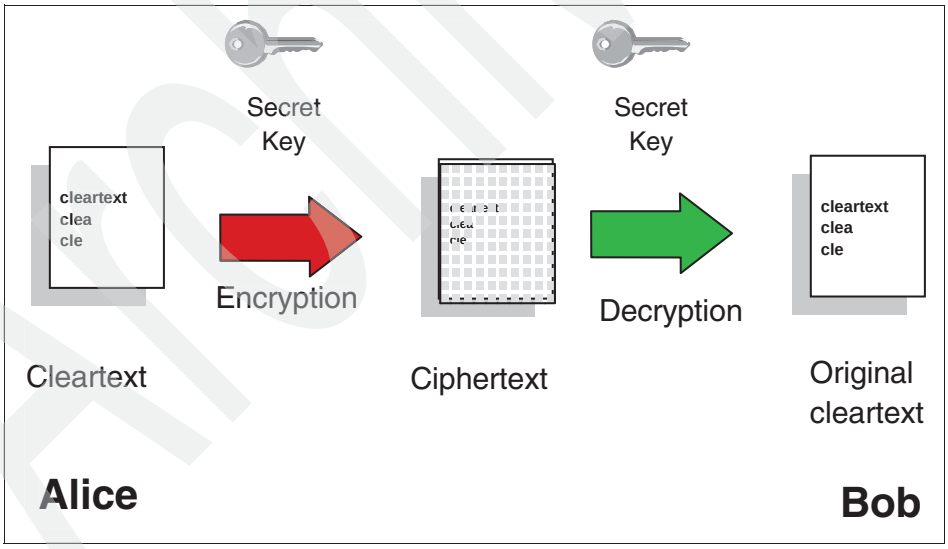


Figure 22-2 Keyed encryption and decryption

Note: It is common in cryptographic literature to denote the first participant in a protocol as Alice and the second one as Bob. They are the “crypto couple.”

Authentication, integrity, and non-repudiation

Encryption provides confidentiality to messages. When communicating over an untrusted medium, such as the Internet, you might also need, in addition to confidentiality:

- Authentication: A method for verifying that the sender of a message is really who he or she claims to be. Any intruder masquerading as someone else is detected by authentication.
- Integrity checking: A method for verifying that a message has not been altered along the communication path. Any tampered message sent by an intruder is detected by an integrity check. As a side effect, communication errors are also detected.
- Non-repudiation: The possibility to prove that the sender has really sent the message. When algorithms providing non-repudiation are used, the sender is not able to later deny the fact that he or she sent the message in question.

22.2.2 Symmetric or secret-key algorithms

Symmetric algorithms are keyed algorithms where the decryption key is the same as the encryption key. These are conventional cryptographic algorithms where the sender and the receiver must agree on the key *before* any secured communication can take place between them. Figure 22-2 on page 778 illustrates a symmetric algorithm. There are two types of symmetric algorithms: *block algorithms*, which operate on the cleartext in blocks of bits, and *stream algorithms*, which operate on a single bit (or byte) of cleartext at a time.

Block ciphers are used in several modes. Electronic Codebook Mode (ECB) is the simplest; each block of cleartext is encrypted independently. Given a block length of 64 bits, there are 264 possible input cleartext blocks, each of them corresponding to exactly one out of 264 possible ciphertext blocks. An intruder might construct a codebook with known cleartext-ciphertext pairs and mount an attack. Because of this vulnerability, the Cipher Block Chaining (CBC) mode is often used, where the result of the encryption of the previous block is used in the encryption of the current block, thus each ciphertext block is dependent not just on the corresponding plaintext block, but on all previous plaintext blocks.

The algorithms often make use of initialization vectors (IVs). These are variables independent of the keys and are good for setting up the initial state of the algorithms.

A well-known block algorithm is the Data Encryption Standard (DES), which was a worldwide standard cipher developed by IBM. DES operates on 64-bit blocks and has a key length of 56 bits, often expressed as a 64-bit number, with every

eighth bit serving as parity bit. From this key, 16 subkeys are derived, which are used in the 16 rounds of the algorithm.

DES produces ciphertexts the same length as the cleartext and the decryption algorithm is exactly the same as the encryption, the only difference being the subkey schedule. These properties make it very suitable for hardware implementations.

DES is becoming obsolete (its origins date back to the early 1970s) and is no longer sufficient as a standard. The most practical attack against it is *brute-force* decryption, with all possible keys, looking for a meaningful result. The problem with DES is the key length. Given enough time and computers, a brute-force attack against the 56-bit key might be feasible. That is why newer modes of DES, called triple-DES, or 3DES, have become popular. With triple-DES, the original DES algorithm is applied in three rounds, with two or three different keys.

Today, DES is still widely used in many forms but has been replaced as a standard by the Advanced Encryption Standard (AES), which is based on a block cipher named Rijndael. The Rijndael cipher is based on a block cipher Square. The Rijndael key length and block size are both variable and can be set to 128, 192, or 256 bits, but the official block size is 128 bits.

Another, block algorithm is the International Data Encryption Algorithm (IDEA). This cipher uses 64-bit blocks and 128-bit keys. It was developed in the early 1990s and aimed to replace DES. It is cryptographically strong and faster than DES. The most significant use of IDEA is in the freeware secure e-mail package Pretty Good Privacy (PGP).

An example of a stream algorithm is A5, which is used to encrypt digital cellular telephony traffic in the GSM standard, widely used in Europe.

The advantage of the symmetric algorithms is their efficiency. They can be easily implemented in hardware. A major disadvantage is the difficulty of key management. A secure way of exchanging the keys must exist, which is often very hard to implement.

22.2.3 Asymmetric or public key algorithms

These algorithms address the major drawback of symmetric ciphers, the requirement of the secure key-exchange channel. The idea is that two different keys should be used: a public key, which, as the name implies, is known to everyone, and a private key, which is to be kept in tight security by the owner. The private key cannot be determined from the public key. A cleartext encrypted with the public key can only be decrypted with the corresponding private key. A cleartext encrypted with the private key can only be decrypted with the

corresponding public key. Therefore, if someone sends a message encrypted with the recipient's public key, it can be read by the intended recipient only. The process is shown in Figure 22-3, where Alice sends an encrypted message to Bob.

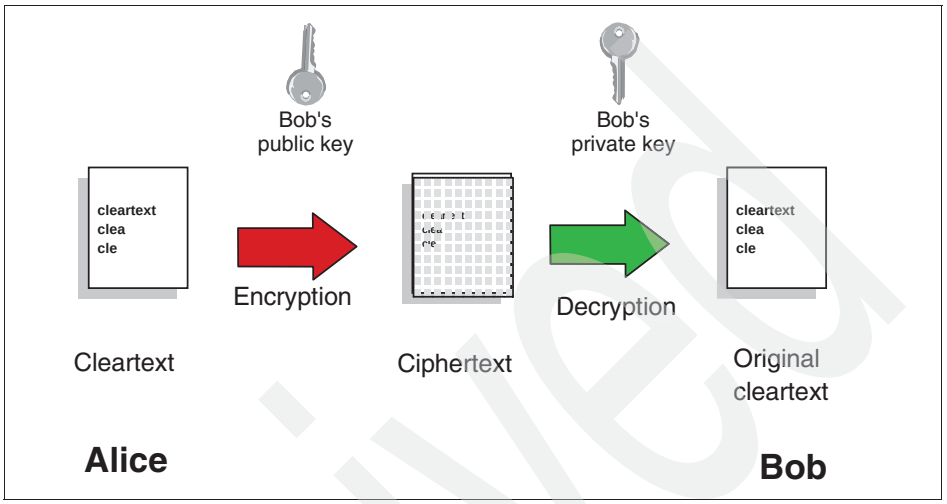


Figure 22-3 Encryption using the recipient's public key

As the public key is available to anyone, privacy is assured without the need for a secure key-exchange channel. Parties that want to communicate retrieve each other's public key.

Authentication and non-repudiation

An interesting property of the public key algorithms is that they can provide authentication. The private key is used for encryption. Because anyone has access to the corresponding public key and can decrypt the message, this provides no privacy. However, it authenticates the message. If you can successfully decrypt it with the claimed sender's public key, the message has been encrypted with the corresponding private key, which is known by the real sender only. Therefore, the sender's identity is verified. Encryption with the private key is used in *digital signatures*. The principle is shown in Figure 22-4 on page 782. Alice encrypts her message with her private key (“signs” it), in order to enable Bob to verify the authenticity of the message.

Archived

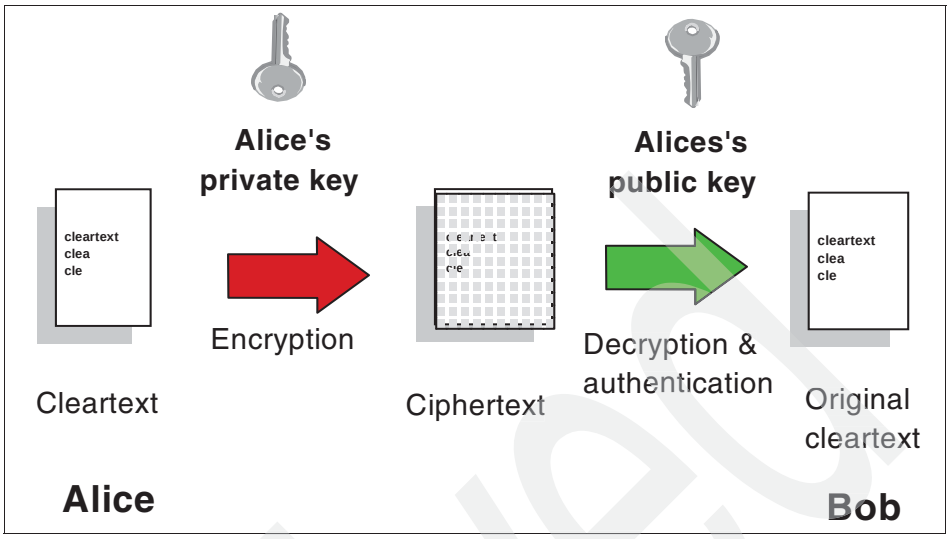


Figure 22-4 Authentication by encrypting with a private key

Going a step further, encrypting with the private key gives non-repudiation, too. The mere existence of such an encrypted message testifies that the originator has really sent it, because only he or she could have used the private key to generate the message. Additionally, if a time stamp is included, the exact date and time can also be proven. There are protocols involving trusted third parties that prevent the sender from using phony time stamps.

Examples of public key algorithms

Algorithms based on public keys can be used for a variety of purposes. Two common applications are:

- Encryption (see “RSA public key algorithm” on page 783).
- Generation of shared keys for use with symmetric key algorithms (see “Diffie-Hellman key exchange” on page 784).

The most popular public key algorithm is the *de facto* standard RSA, named after its three inventors: Ron Rivest, Adi Shamir, and Leonard Adleman. The security of RSA relies on the difficult problem of factoring large numbers. The public and private keys are functions of two very large (200 digits or even more) prime numbers. Given the public key and the ciphertext, an attack is successful if it can factor the product of the two primes. RSA has resisted many years of extensive attacks. As computing power grows, keeping RSA secure is a matter of increasing the key length (unlike DES, where the key length is fixed).

Another public key algorithm, the very first ever invented, is *Diffie-Hellman*. This is a key exchange algorithm; that is, it is used for securely establishing a shared secret over an insecure channel. The communicating parties exchange public information from which they derive a key. An eavesdropper cannot reconstruct the key from the information that went through the insecure channel. More precisely, the reconstruction is computationally infeasible. The security of Diffie-Hellman relies on the difficulty of calculating discrete logarithms in finite fields. After the shared secret has been established, it can then be used to derive keys for use with symmetric key algorithms such as DES.

Diffie-Hellman makes the secure derivation of a shared secret key possible, but it does not authenticate the parties. For authentication, another public key algorithm must be used, such as RSA.

Unfortunately, public key algorithms, while providing for easier key management, privacy, authentication, and non-repudiation, also have some disadvantages. The most important one is that they are slow and difficult to implement in hardware. For example, RSA is 100 to 10,000 times slower than DES, depending on implementation. Because of this, public key algorithms generally are not used for bulk encryption. Their most important use is key exchange and authentication. Another notable disadvantage is that they are susceptible to certain cryptanalytic attacks to which symmetric algorithms are resistant.

Therefore, a good cryptographic system (*cryptosystem*) makes use of both worlds. It uses public key algorithms in the session establishment phase for authentication and key exchange, and then a symmetric one for encrypting the consequent messages.

For the interested reader, we give more detailed information of the two most important asymmetric algorithms, which involve modular arithmetic. An arithmetic operation modulo m means that the result of that operation is divided by m and the remainder is taken. For example: 3 * 6 mod 4 = 2, since 3 * 6 = 18 and dividing 18 by 4 gives us 2 as the remainder.

RSA public key algorithm

RSA is used in the ISAKMP/Oakley framework as one of the possible authentication methods. The principle of the RSA algorithm is as follows:

1. Take two large primes, p and q.
2. Find their product n = pq; n is called the modulus.
3. Choose a number, e, less than n and relatively prime to (p-1)(q-1), which means that e and (p-1)(q-1) have no common factor other than 1.
4. Find its inverse, d mod (p-1)(q-1), which means that ed = 1 mod (p-1)(q-1).

e and d are called the public and private exponents, respectively. The public key is the pair (n,e); the private key is d. The factors p and q must be kept secret or destroyed.

A simplified example of RSA encryption is:

1. Suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating:

   $c = m^e \bmod n$

   Where e and n are Bob's public key.
2. Alice sends c to Bob.
3. To decrypt, Bob exponentiates:

   $m = c^d \bmod n$

   And recovers the original message; the relationship between e and d ensures that Bob correctly recovers m. Because only Bob knows d, only Bob can decrypt the ciphertext.

A simplified example of RSA authentication is:

1. Suppose Alice wants to send a signed message, m, to Bob. Alice creates a digital signature s by exponentiating:

   $s = m^d \bmod n$

   Where d and n belong to Alice's private key.
2. She sends s and m to Bob.
3. To verify the signature, Bob exponentiates and checks if the result, compares to m:

   $m = s^e \bmod n$

   Where e and n belong to Alice's public key.

Diffie-Hellman key exchange

The Diffie-Hellman key exchange is a crucial component of the ISAKMP/Oakley framework. In the earliest phase of a key negotiation session, there is no secure channel in place. The parties derive shared secret keys using the Diffie-Hellman algorithm. These keys will be used in the next steps of the key negotiation protocol.

The following steps outline the algorithm:

1. The parties (Alice and Bob) share two public values, a modulus m and an integer g. m is a large prime number.

2. Alice generates a large random number a and computes:

   $X = g^a \bmod m$

3. Bob generates a large random number b and computes:

   $Y = g^b \bmod m$

4. Alice sends X to Bob.
5. Bob computes:

   $K1 = X^b \bmod m$

6. Bob sends Y to Alice.
7. Alice computes:

   $K2 = Y^a \bmod m$

Both K1 and K2 are equal to $g^{ab} \bmod m$. This is the shared secret key. No one is able to generate this value without knowing a or b. The security of the exchange is based on the fact that is extremely difficult to inverse the exponentiation performed by the parties. (In other words, to calculate discrete logarithms in finite fields of size m.) Similar to RSA, advances in adversary computing power can be countered by choosing larger initial values, in this case a larger modulus m.

See 22.4.5, “Internet Key Exchange (IKE) protocol” on page 829 for more details about how ISAKMP/Oakley uses Diffie-Hellman exchanges.

22.2.4 Hash functions

Hash functions (also called message digests) are fundamental to cryptography. A hash function is a function that takes variable-length input data and produces fixed length output data (the hash value), which can be regarded as the “fingerprint” of the input. That is, if the hashes of two messages match, it is highly probable that the messages are the same.

Cryptographically useful hash functions must be *one-way*, which means that they should be easy to compute, but infeasible to reverse. An everyday example of a one-way function is mashing a potato; it is easy to do, but once mashed, reconstructing the original potato is rather difficult.

A good hash function must also be *collision-resistant*. It must be hard to find two different inputs that hash to the same value. Because any hash function maps an input set to a smaller output set, theoretically it is possible to find collisions. The point is to provide a unique digital “fingerprint” of the message that identifies it with high confidence, much like a real fingerprint identifying a person.

A hash function that takes a key as a second input parameter and its output depends on both the message and the key is called a *message authentication code (MAC)*, as shown in Figure 22-5.

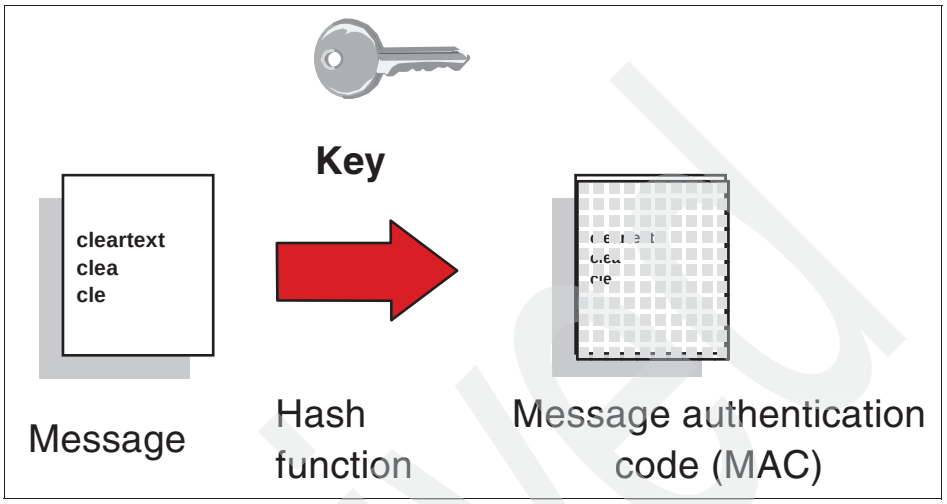


Figure 22-5 Generating a message authentication code (MAC)

Put simply, if you encrypt a hash, it becomes a MAC. If you add a secret key to a message, and then hash the concatenation, the result is a MAC. Both symmetric and asymmetric algorithms can be used to generate MACs.

Hash functions are primarily used to assure integrity and authentication:

- The sender calculates the hash of the message and appends it to the message.
- The recipient calculates the hash of the received message and then compares the result with the transmitted hash.
- If the hashes match, the message was not tampered with.
- If the encryption key (symmetric or asymmetric) is only known by a trusted sender, a successful MAC decryption indicates that the claimed and actual senders are identical.

Archive

See Figure 22-6 for an illustration of the procedure. The Message* and MAC* notations reflect the fact that the message might have been altered while crossing the untrusted channel.

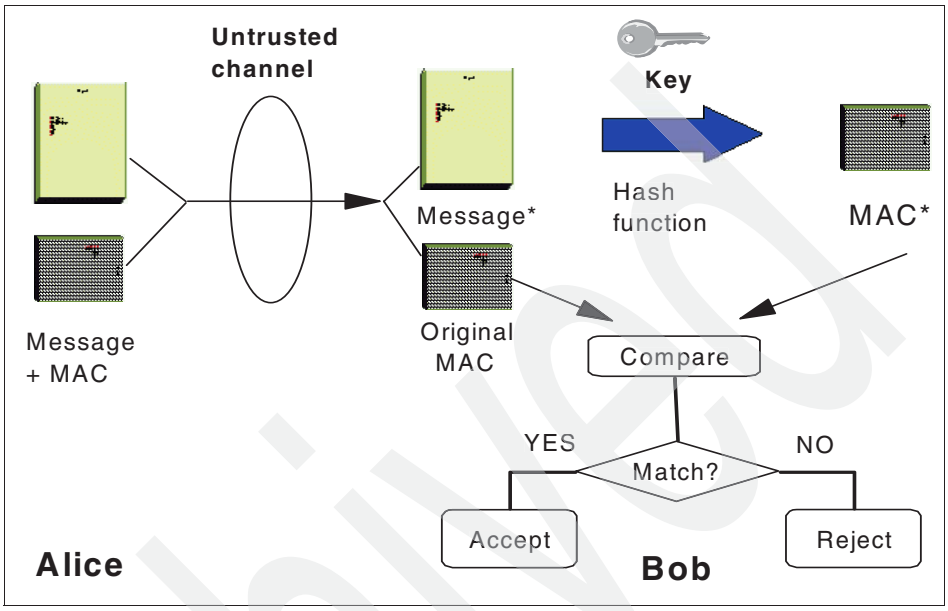


Figure 22-6 Checking integrity and authenticity with MAC

You might argue that the same result can be obtained with any kind of encryption, because if an intruder modifies an encrypted message, the decryption will result in nonsense, thus tampering can be detected. The answer is that many times only integrity, authentication, or both are needed, maybe with encryption on some of the fields of the message. Also encryption is very processor-intensive. Examples include the personal banking machine networks, where only the PINs are encrypted. However, MACs are widely used. Encrypting all the messages in their entirety would not yield noticeable benefits and performance would dramatically decrease.

The encryption of a hash with the private key is called a *digital signature*. It can be thought of as a special MAC. Using digital signatures instead of encrypting the whole message with the private key leads to considerable performance gains and a remarkable new property. The authentication part can be decoupled from the document itself. This property is used, for example, in the Secure Electronic Transactions (SET) protocol.

The encryption of a secret key with a public key is called a *digital envelope*. This is a common technique used to distribute secret keys for symmetric algorithms.

Examples of hash functions

The most widely used hash functions are MD5 and Secure Hash Algorithm 1 (SHA-1). MD5 was designed by Ron Rivest (co-inventor of RSA). SHA-1 is largely inspired from MD5 and was designed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) for use with the Digital Signature Standard (DSS). MD5 produces a 128-bit hash, while SHA-1 produces a 160-bit hash. Both functions encode the message length in their output. SHA-1 is regarded as more secure, because of the larger hashes it produces.

Neither MD5 nor SHA-1 takes a key as an input parameter. Therefore, in their original implementation, they cannot be used for MAC calculation. However, for this purpose, it is easy to concatenate a key with the input data and apply the function to the result.

Note: In practice, for example, in IPSec, more sophisticated schemes are often used.

Keyed MD5 and keyed SHA-1

Using MD5 and SHA-1 in keyed mode is simple. The shared secret key and the data to be protected are both input to the hash algorithm. In the following IPSec example, the datagram is combined with the key, and the output hash value is placed in the Authentication Data field of the AH header, as shown in Figure 22-7.

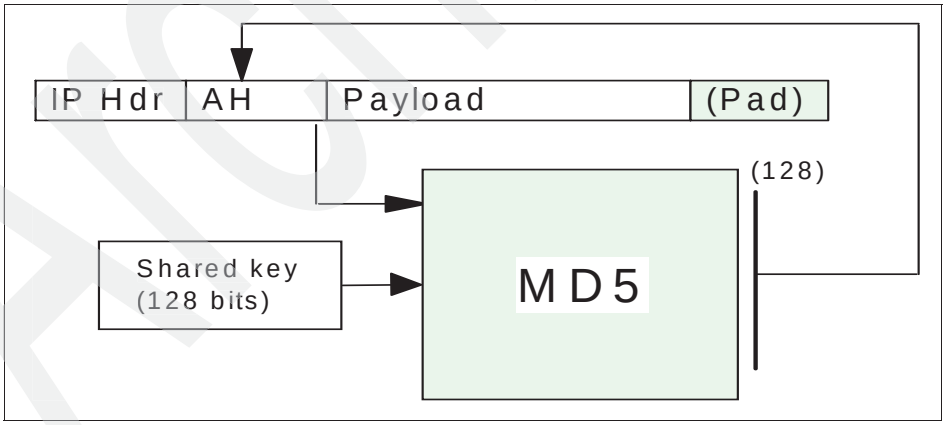


Figure 22-7 Keyed MD5 processing

Keyed SHA-1 operates in the same way, the only difference being the larger 160-bit hash value.

HMAC-MD5-96 and HMAC-SHA-1-96

A stronger method is the Hashed Message Authentication Code (HMAC), proposed by IBM. HMAC itself is not a hash function, rather a cryptographically strong way to use a specific hash function for MAC calculation.

To show how HMAC works, consider MD5 as an example. The base function is applied twice in succession. In the first round, the input to MD5 is the shared secret key and the datagram. The 128-bit output hash value and the key are input again to the hash function in the second round. The left-most 96 bits of the resulting hash value are used as the MAC for the datagram. See Figure 22-8 for an illustration.

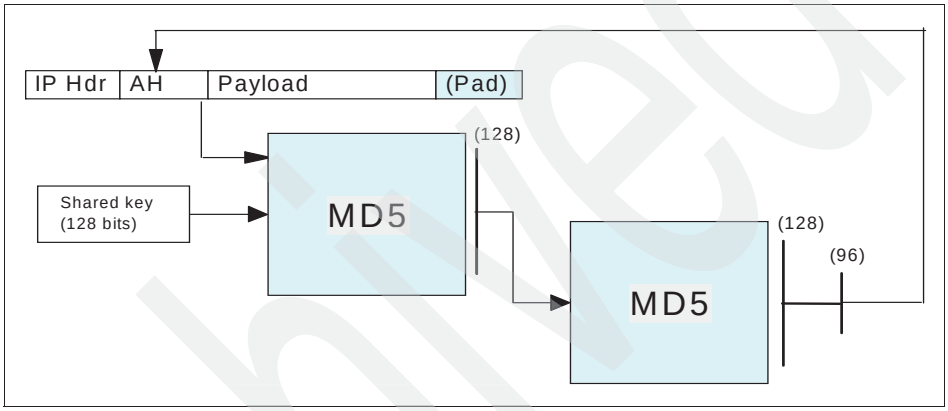


Figure 22-8 HMAC-MD5-96 processing

HMAC-SHA-1-96 operates in the same way, except that the intermediary results are 160 bits long.

Archived

Digital Signature Standard (DSS)

As mentioned previously, a hash value encrypted with the private key is called a *digital signature* and is illustrated in Figure 22-9.

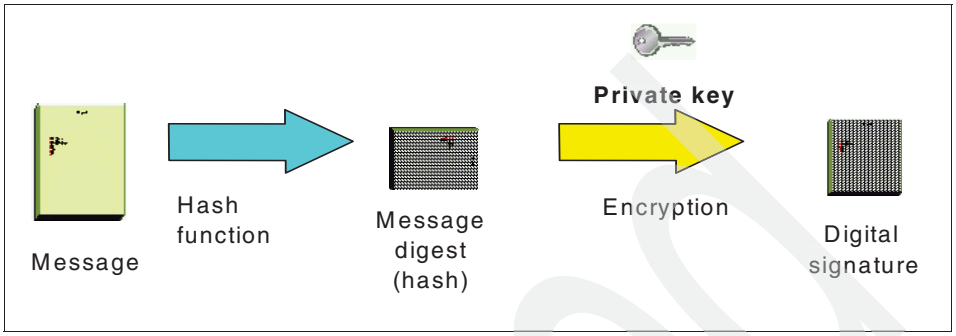


Figure 22-9 Generating a digital signature

One authentication method that can be used with ISAKMP/Oakley is DSS, which was selected by NIST and NSA to be the digital authentication standard of the U.S. government. The standard describes the Digital Signature Algorithm (DSA) used to sign and verify signatures of message digests produced with SHA-1.

The following steps provide a brief description of DSA:

1. Choose a large prime number, p, usually between 512 and 1024 bits long.
2. Find a prime factor q of (p-1), 160 bits long.
3. Compute:

   $g=h^{(p-1)/q} \bmod p$

   Where h is a number less than (p-1) and the following is true:

   $h^{(p-1)/q}>1$
4. Choose another number x, less than q, as the sender's private key.
5. Compute:

   $y=g^{x} \bmod p$

   And use that as the sender's public key. The pair (x,y) is sometimes referred to as the long-term key pair.

6. The sender signs the message as follows:

   a. Generate a random number, k, less than q.
   b. Compute:

      $r=(g^k \bmod p) \bmod q$
      $s=(k^{-1}(SHA1(m)+xr)) \bmod q$

      The pair (k,r) is sometimes referred to as the per-session key pair, and the signature is represented by the pair (r,s).

7. The sender sends (m,r,s).
8. The receiver verifies the signature as follows:

   Compute:

   $w=s^{-1} \bmod q$
   $u1=(SHA1(m)*w) \bmod q$
   $u2=(rw) \bmod q$
   $v=((g^{u1}y^{u2}) \bmod p) \bmod q$

9. If v=r, the signature is verified.

22.2.5 Digital certificates and certification authorities

As mentioned in “Authentication and non-repudiation” on page 781, with public key cryptography, the parties retrieve each other's public key. However, there are security exposures here. An intruder can replace some real public keys with his or her own public key, and then mount a so-called *man-in-the-middle attack*.

For example, the intruder places himself between Alice and Bob. He can trick Bob by sending him one of his own public keys as though it were Alice's. The same applies to Alice. She thinks she uses Bob's public key, but she actually uses the intruder's. So, the clever intruder can decrypt the confidential traffic between the two and remain undetected. For example, a message sent by Alice and encrypted with “Bob's” public key arrives at the intruder, who decrypts it, learns its content, then re-encrypts it with Bob's real public key. Bob has no way to realize that Alice is using a phony public key.

An intruder can also use impersonation, claiming to be somebody else, for example, an online shopping mall, fooling innocent shoppers.

The solution to these serious threats is the *digital certificate*. A digital certificate is a file that binds an identity to the associated public key. This binding is validated by a trusted third party, the *certification authority (CA)*. A digital certificate is signed with the private key of the certification authority, so it can be authenticated. It is only issued after a verification of the applicant. Apart from the

public key and identification, a digital certificate usually contains other information too, such as:

- Date of issue
- Expiration date
- Miscellaneous information from the issuing CA (for example, serial number)

Note: There is an international standard in place for digital certificates: The ISO X.509 protocols.

The parties retrieve each other's digital certificate and authenticate it using the public key of the issuing certification authority. They have confidence that the public keys are real, because a trusted third party vouches for them. This helps protect against both man-in-the-middle and impersonation attacks.

It is easy to imagine that one CA cannot cover all needs. What happens when Bob's certificate is issued by a CA unknown to Alice? Can she trust that unknown authority? Well, this is entirely her decision, but to make life easier, CAs can form a hierarchy, often referred to as the *trust chain*. Each member in the chain has a certificate signed by its superior authority. The higher the CA is in the chain, the tighter security procedures are in place. The root CA is trusted by everyone and its private key is top secret.

Alice can traverse the chain upward until she finds a CA that she trusts. The traversal consists of verifying the subordinate CA's public key and identity using the certificate issued to it by the superior CA.

When a trusted CA is found in the chain, Alice is assured that Bob's issuing CA is trustworthy. This is all about delegation of trust. We trust your identity card if somebody who we trust signs it. And if the signer is unknown to us, we can go upward and see who signs for the signer, and so on.

An implementation of this concept is in the SET protocol, where the major credit card brands operate their own CA hierarchies that converge to a common root. Lotus® Notes® authentication, as another example, is also based on certificates, and it can be implemented using hierarchical trust chains. PGP also uses a similar approach, but its trust chain is based on persons and it is a distributed Web rather than a strict hierarchical tree.

22.2.6 Random-number generators

An important component of a cryptosystem is the random-number generator. Many times random session keys and random initialization variables (often referred to as initialization vectors) are generated. For example, DES requires an

explicit initialization vector and Diffie-Hellman relies on picking random numbers which serve as input for the key derivation.

The quality, that is the randomness of these generators, is more important than you might think. The ordinary random function provided with most programming language libraries is good enough for games, but not for cryptography. Those random-number generators are rather predictable; if you rely on them, be prepared for happy cryptanalysts finding interesting correlations in your encrypted output.

The fundamental problem faced by the random-number generators is that the computers are ultimately deterministic machines, so real random sequences cannot be produced. As John von Neumann ironically said: “Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.” That's why the term *pseudorandom generator* is more appropriate.

Cryptographically strong pseudorandom generators must be unpredictable. It must be computationally infeasible to determine the next random bit, even with total knowledge of the generator.

A common practical solution for pseudorandom generators is to use hash functions. This approach provides sufficient randomness and it can be efficiently implemented. Military-grade generators use specialized devices that exploit the inherent randomness in physical phenomena. An interesting solution can be found in the PGP software. The initial seed of the pseudorandom generator is derived from measuring the time elapsed between the keystrokes of the user.

22.2.7 Export/import restrictions on cryptography

U.S. export regulations changed on January 14, 2000 with the publication of new regulations in the Federal Register. These regulations make it easier for United States companies and individuals to export strong encryption. Some of the changes include:

- “Retail” encryption products are widely exportable to all but certain “terrorist” nations though still subject to a government review and reporting requirements.
- Non-retail products are also exportable, subject to similar requirements, to most non-government users.
- Encryption products with less than 64-bits are freely exportable.
- Some non-proprietary source code is exportable to most countries after notice to the government.

In September 1998, the White House announced further liberalization of U.S. export restrictions on cryptographic material and key recovery requirements, which can be summarized as follows:

- The key recovery requirement for export of 56-bit DES and equivalent products is eliminated. This includes products that use 1024-bit asymmetric key exchanges together with 56-bit symmetric key algorithms.
- Export of unlimited strength encryption (for example, 3DES) under license exceptions (with or without key recovery) is now broadened to include others besides the financial industry for 45 countries. This includes subsidiaries of U.S firms, insurance, health and medical (excluding biochemical and pharmaceutical manufacturers), and online merchants for the purpose of securing online transactions (excluding distributors of items considered munitions).

  For the latter, recoverable products will be granted exceptions world wide (excluding terrorist countries) without requiring a review of foreign key recovery agents.
- Export of recoverable products will be granted to most commercial firms, for a broad range of countries, in the major commercial markets (excluding items on the U.S. munitions list).
- Export licenses to end users may be granted on a case-by-case basis.

More information can be obtained from the U.S. Department of Commerce:

`http://www.bis.doc.gov/Encryption/Default.htm`

According to the law in France, any product capable of enciphering/deciphering user data must be granted a license from the French government before being marketed. Clients need to be authorized to use such products on a case-by-case basis. In reality, two major and useful exceptions exist:

- Routinely, licenses are granted that allow banks to use DES products on a global basis (no case-by-case authorization required).
- Routinely, global licenses are granted that allow anybody to use weak encryption (RC2/RC4 with 40-bit keys).

22.3 Firewalls

Firewalls have significant functions in an organization's security policy. Therefore, it is important to understand these functions and apply them to the network properly. This chapter explains the firewall concept, network security, firewall components, and firewall examples.

22.3.1 Firewall concept

A firewall is a system (or group of systems) that enforces a security policy between a secure internal network and an untrusted network such as the Internet. Firewalls tend to be seen as a protection between the Internet and a private network. But generally speaking, a firewall should be considered as a means to divide the world into two or more networks: one or more secure networks and one or more non-secure networks. See Figure 22-10.

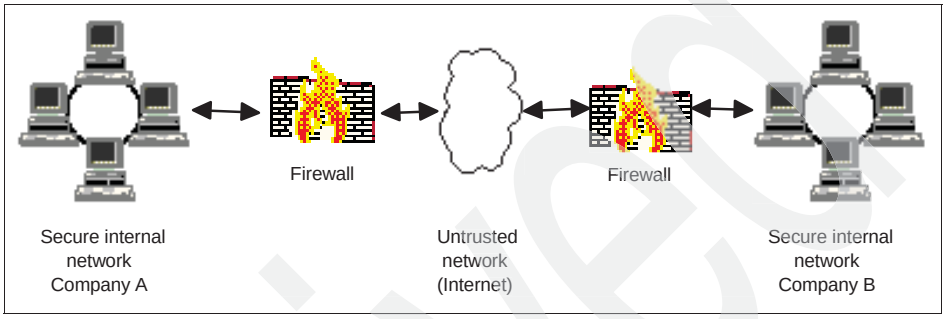


Figure 22-10 A firewall illustration

A firewall can be a PC, a router, a midrange, a mainframe, a UNIX workstation, or a combination of these that determines which information or services can be accessed from the outside and who is permitted to use the information and services from outside. Generally, a firewall is installed at the point where the secure internal network and untrusted external network meet, which is also known as a *choke point*.

In order to understand how a firewall works, consider the network to be a building to which access must be controlled. The building has a lobby as the only entry point. In this lobby, receptionists welcome visitors, security guards watch visitors, video cameras record visitor actions, and badge readers authenticate visitors who enter the building.

Although these procedures can work well to control access to the building, if an unauthorized person succeeds in entering, there is no way to protect the building against this intruder's actions. However, if the intruder's movements are monitored, it can be possible to detect any suspicious activity.

Similarly, a firewall is designed to protect the information resources of the organization by controlling the access between the internal secure network and the untrusted external network (see Figure 22-11 on page 796). However, it is important to note that even if the firewall is designed to permit the trusted data to pass through, deny the vulnerable services, and prevent the internal network from outside attacks, a newly created attack can penetrate the firewall at any

time. The network administrator must examine all logs and alarms generated by the firewall on a regular basis. Otherwise, it is generally not possible to protect the internal network from outside attacks.

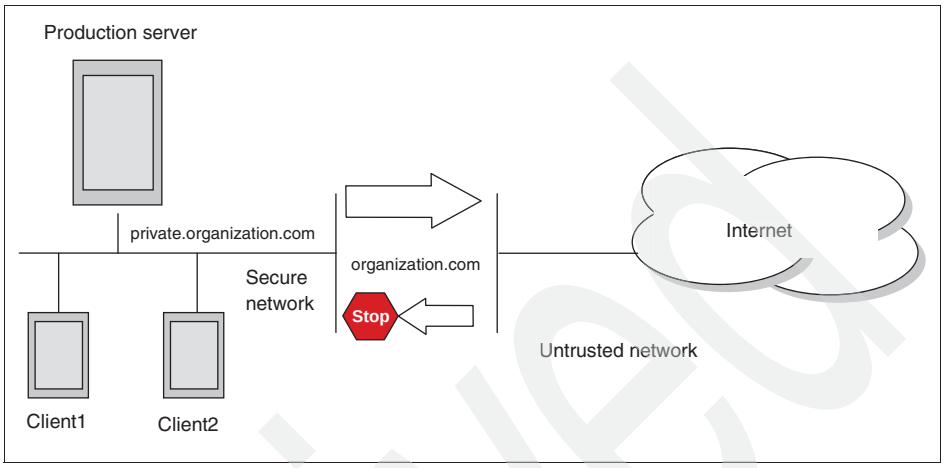


Figure 22-11 A firewall controls traffic between the secure network and the Internet

22.3.2 Components of a firewall system

As mentioned previously, a firewall can be a PC, a midrange, a mainframe, a UNIX workstation, a router, or combination of these. Depending on the requirements, a firewall can consist of one or more of the following functional components:

- Packet-filtering router
- Application-level gateway (proxy)
- Circuit-level gateway

Each of these components has different functions and shortcomings. Generally, in order to build an effective firewall, these components are used together.

Archived

Packet-filtering router

Most of the time, packet-filtering is accomplished by using a router that can forward packets according to filtering rules. When a packet arrives at the packet-filtering router, the router extracts certain information from the packet header and makes decisions according to the filter rules as to whether the packet will pass through or be discarded (see Figure 22-12). The following information can be extracted from the packet header:

- Source IP address
- Destination IP address
- TCP/UDP source port
- TCP/UDP destination port
- ICMP message type
- Encapsulated protocol information (TCP, UDP, ICMP, or IP tunnel)

The packet-filtering rules are based on the network security policy (see 22.1.4, “Network security policy” on page 776). Therefore, packet-filtering is done by using these rules as input. When determining the filtering rules, outside attacks must be taken into consideration, as well as service level restrictions and source/destination level restrictions.

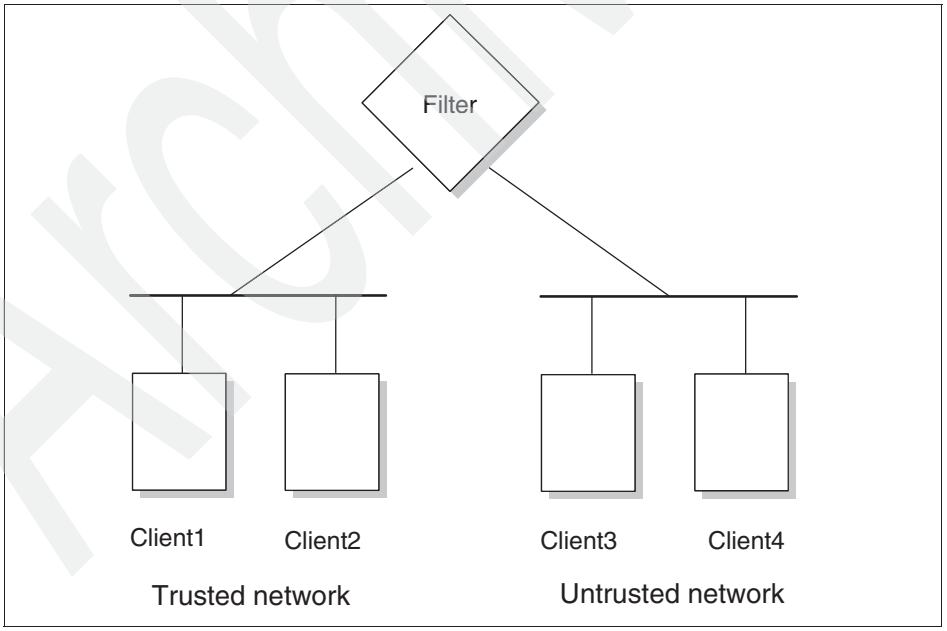


Figure 22-12 Packet-filtering router

Archived

Service level filtering

Because most services use well-known TCP/UDP port numbers, it is possible to allow or deny services by using related port information in the filter. For example, an FTP server listens for connections on TCP port 21, and for a non-passive mode client, makes outbound data connections from port 20. Therefore, to permit FTP connections to pass through to a secure network, the router can be configured to permit packets that contain 20 and 21 as the TCP port in its header. However, there are some applications, such as NFS, that use RPC and use different ports for each connection. Allowing these kind of services might cause security problems.

Source/destination level filtering

The packet-filtering rules allow a router to permit or deny a packet according to the destination or the source information in the packet header. In most cases, if a service is available, only that particular server is permitted to outside users. Other packets that have another destination or no destination information in their headers are discarded.

Advanced filtering

As mentioned previously (see 22.1.1, “Common attacks against security” on page 772), there are different types of attacks that threaten the privacy and network security. Some of them can be discarded by using advanced filtering rules such as checking IP options, fragment offset, and so on.

Packet-filtering limitations

Packet-filtering rules are sometimes very complex. When there are exceptions to existing rules, it becomes much more complex. Although there are a few testing utilities available, it is still possible to leave some holes in the network security. Packet filters do not provide an absolute protection for a network. For some cases, it might be necessary to restrict some set of information (for example, a command) from passing through to the internal secure network. It is not possible to control the data with packet filters because they are not capable of understanding the contents of a particular service. For this purpose, an application-level control is required.

Application-level gateway (proxy)

An application-level gateway is often referred to as a *proxy*. An application-level gateway provides higher-level control on the traffic between two networks in that the contents of a particular service can be monitored and filtered according to the

network security policy. Therefore, for any desired application, the corresponding proxy code must be installed on the gateway in order to manage that specific service passing through the gateway (see Figure 22-13).

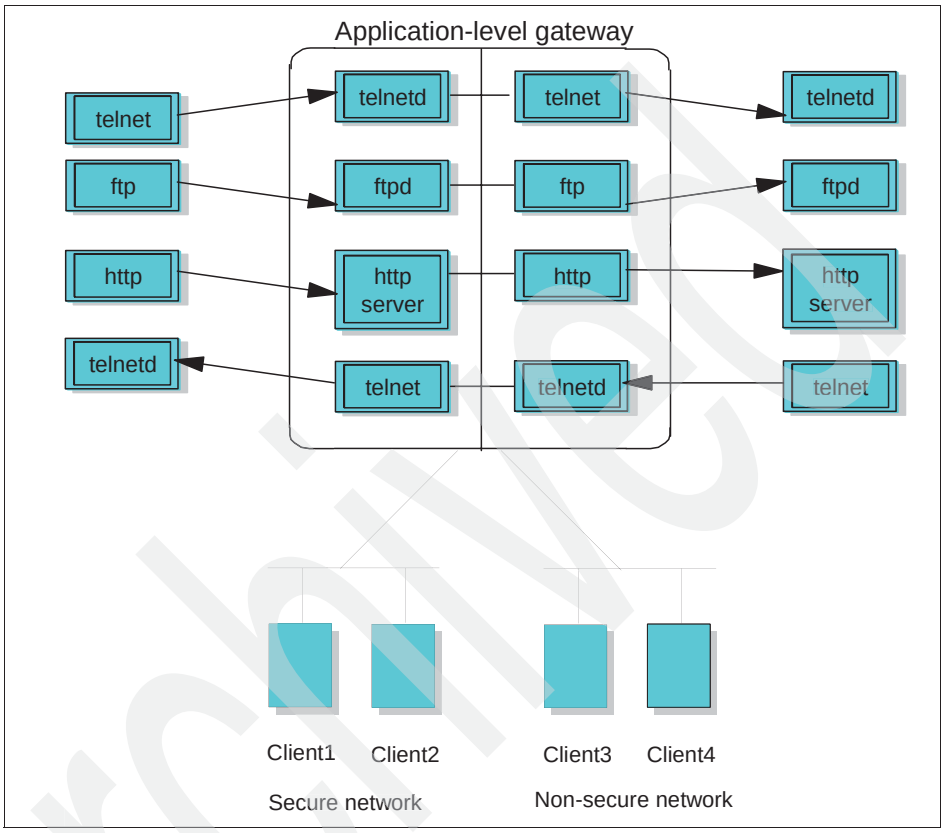


Figure 22-13 Application-level gateway

A proxy acts as a server to the client and as a client to the destination server. A virtual connection is established between the client and the destination server. Though the proxy seems to be *transparent* from the point of view of the client and the server, the proxy is capable of monitoring and filtering any specific type of data, such as commands, before sending it to the destination. For example, an FTP server is permitted to be accessed from outside. In order to protect the server from any possible attacks, the FTP proxy in the firewall can be configured to deny PUT and MPUT commands.

A proxy server is an application-specific relay server that runs on the host that connects a secure and a non-secure network. The purpose of a proxy server is to control exchange of data between the two networks at an application level instead of an IP level. By using a proxy server, it is possible to disable IP routing

between the secure and the non-secure network for the application protocol the proxy server is able to handle, but still be able to exchange data between the networks by relaying it in the proxy server. Figure 22-14 shows an FTP proxy server.

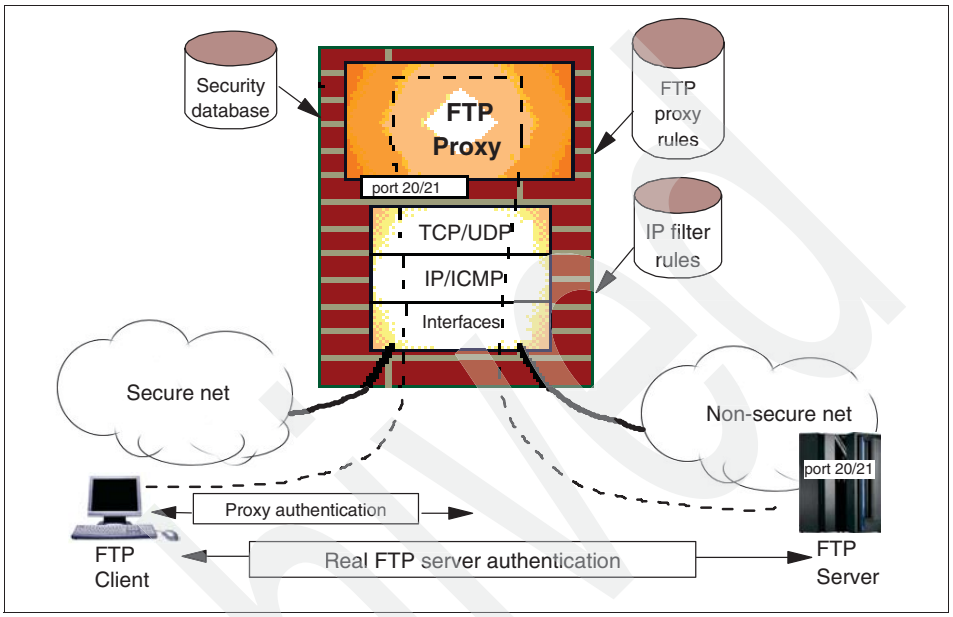


Figure 22-14 FTP proxy server

Note that in order for any client to be able to access the proxy server, the client software must be specifically modified. In other words, the client and server software must support the proxy connection. In the previous example, the FTP client must authenticate itself to the proxy first. If it is successfully authenticated, the FTP session starts based on the proxy restrictions. Most proxy server implementations use more sophisticated authentication methods such as security ID cards. This mechanism generates a unique key that is not reusable for another connection. Two security ID cards are supported by IBM Firewall: the SecureNet card from Axent and the SecureID card from Security Dynamics.

Compared with IP filtering, application-level gateways provide much more comprehensive logging based on the application data of the connections. For example, an HTTP proxy can log the URLs visited by users. Another feature of application-level gateways is that they can use strong user authentication. For

example, when using FTP and Telnet services from the unsecure network, users can be forced to authenticate themselves to the proxy. Figure 22-15 shows a proxy server TCP segment flow example.

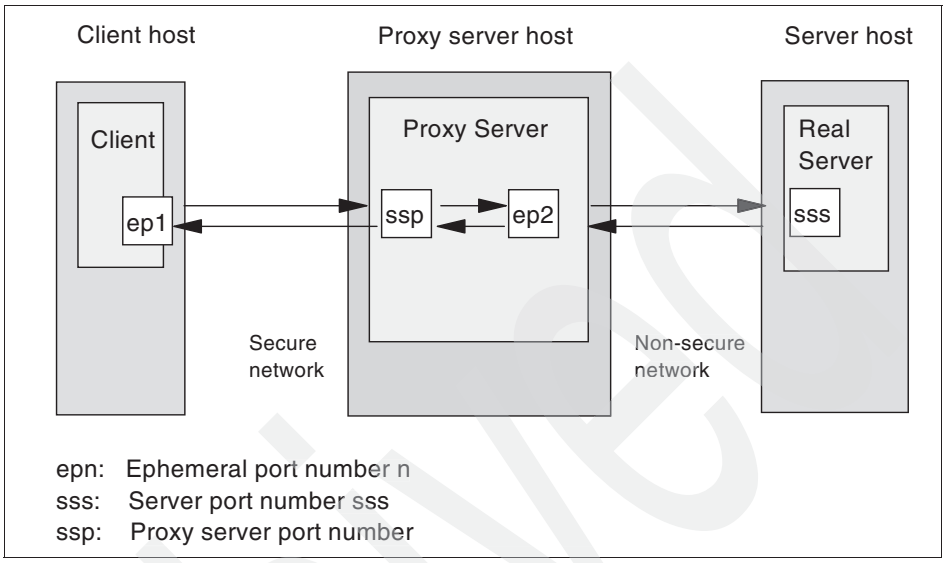


Figure 22-15 Proxy server TCP segment flow

Application-level gateway limitations

A disadvantage of application-level gateways is that, in order to achieve a connection through a proxy server, the client software must be changed to support that proxy service. This can sometimes be achieved by some modifications in user behavior rather than software modification. For example, to connect to a Telnet server over a proxy, the user usually has to be authenticated by the proxy server then by the destination Telnet server. This requires two user steps to make a connection rather than one. However, a modified Telnet client can make the proxy server transparent to the user by specifying the destination host rather than proxy server in the Telnet command.

An example: FTP proxy server

Most of the time, in order to use the FTP proxy server, users must have a valid user ID and password. On UNIX systems, users also must be defined as users of the UNIX system.

FTP can be used in one of two modes:

- Normal mode
- Passive mode

In normal mode, the FTP client first connects to the FTP server port 21 to establish a control connection. When data transfer is required (for example, as the result of a DIR, GET, or PUT command), the client sends a PORT command to the server instructing the server to establish a data connection from the server's data port (port 20) to a specified ephemeral port number on the client host.

In an FTP proxy server situation, normal mode means that we have to allow inbound TCP connections from the non-secure network to the FTP proxy host. Notice in Figure 22-16 how a connection is established from the FTP server port 20 in the non-secure network to the FTP proxy server's ephemeral port number. To allow this to happen, IP filtering rules are used that allow inbound connection requests from port 20 to an ephemeral port number on the FTP proxy host. This is normally not an IP filter rule. It is sometimes better to add a custom filter rule configuration, because it would allow a cracker to run a program on port 20 and scan all the port numbers above 1023, which, in its simplest form, might result in a denial-of-service situation. Some firewalls handle this correctly by building a table of outgoing FTP requests and matching up the corresponding incoming data transfer request.

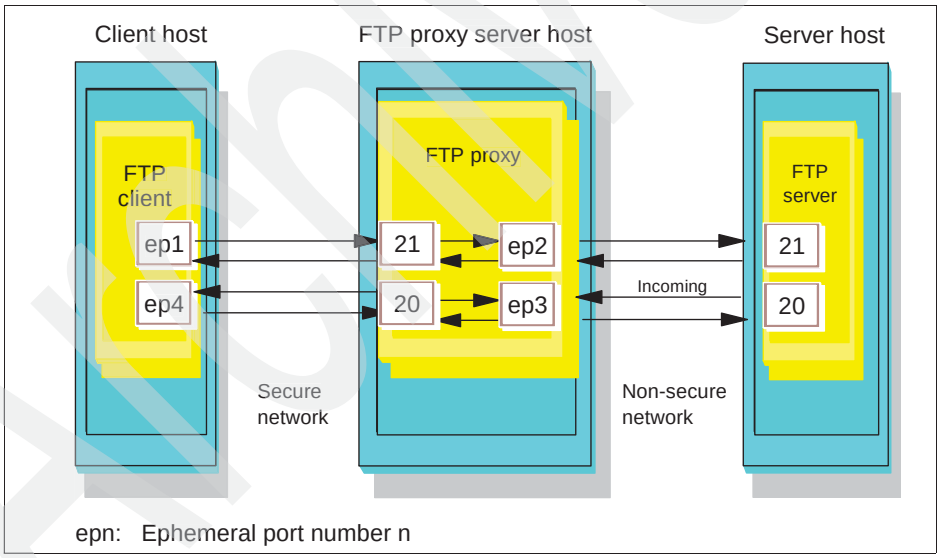


Figure 22-16 Normal mode FTP proxy

A much more firewall-friendly mode is the passive mode of operation, as shown in Figure 22-17. This mode has been dubbed a firewall-friendly FTP and is described in RFC 1579 – Firewall-Friendly FTP.

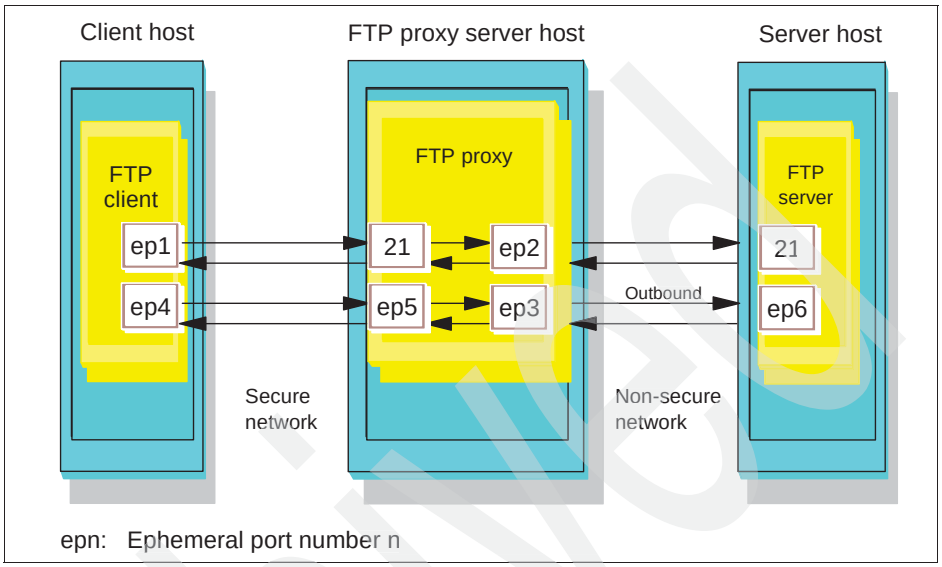


Figure 22-17 Passive mode FTP proxy (firewall-friendly FTP)

In passive mode, the FTP client again establishes a control connection to the server's port 21. When data transfer has to start, the client sends a PASV command to the server. The server responds with a port number for the client to contact, in order to establish the data connection, and the client then initiates the data connection.

In this setup, to establish connections to both port 21 and any ephemeral port number in the non-secure network, an ephemeral port number is used on the FTP proxy host. Here, we do not need a rule that allows inbound connections to ephemeral port numbers, because we are now connecting outward.

Circuit-level gateway

A circuit-level gateway relays TCP connections and does not provide any extra packet processing or filtering. Some circuit-level gateways can handle UDP packets. A circuit-level gateway can be said to be a special type of application-level gateway. This is because the application-level gateway can be configured to pass all information after the user is authenticated, just as the

circuit-level gateway (see Figure 22-18 on page 805). However, in practice, there are significant differences between them, such as:

- Circuit-level gateways can handle several TCP/IP applications, as well as UDP applications, without any extra modifications on the client side for each application. Therefore, this makes circuit-level gateways a good choice to satisfy user requirements.
- Circuit-level gateways do not provide packet processing or filtering. Therefore, a circuit-level gateway is generally referred to as a *transparent* gateway.
- Application-level gateways have a lack of support for UDP.
- Circuit-level gateways are often used for outbound connections, while application-level gateways (proxy) are used for both inbound and outbound connections. Generally, when using both types combined, circuit-level gateways can be used for outbound connections and application-level gateways can be used for inbound connections to satisfy both security and user requirements.

Circuit-level gateways can sometimes handle incoming UDP packets or TCP connections. However, a client on the secure side must inform the gateway to expect such packets. SOCKS v5 has this capability.

A well-known example of a circuit-level gateway is SOCKS (refer to 22.5, “SOCKS” on page 846 for more information). Because the data that flows over SOCKS is not monitored or filtered, a security problem can arise. To minimize security problems, trusted services and resources need to be used on the outside network (untrusted network).

Archive

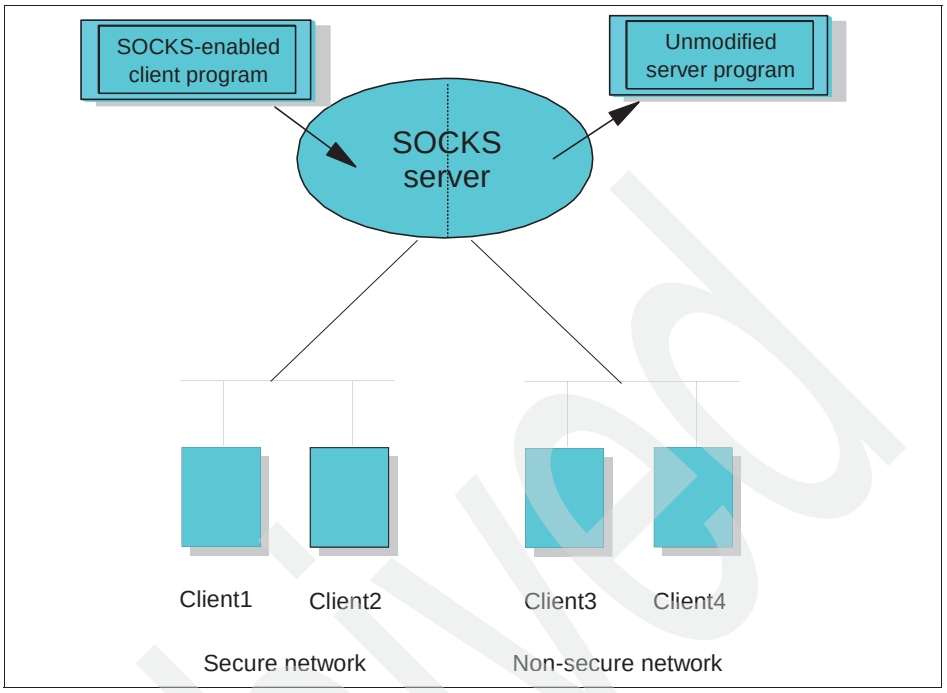


Figure 22-18 Circuit-level gateway

22.3.3 Types of firewalls

A firewall consists of one or more software elements that run on one or more hosts. The hosts can be general purpose computer systems or specialized such as routers. There are four important examples of firewalls. These are:

- Packet-filtering firewall
- Dual-homed gateway firewall
- Screened host firewall
- Screened subnet firewall

Packet-filtering firewall

The packet-filtering firewall is commonly used because it is inexpensive (see Figure 22-19 on page 806). The firewall is just a router sitting between the external network and the internal secure network. Packet-filtering rules are defined to permit or deny traffic (see “Packet-filtering router” on page 797).

Archived

Generally, a packet-filtering firewall is configured to deny any service if it is not explicitly permitted. Although this approach prevents some potential attacks, the firewall is still open to attacks that result from improper filter rule configurations.

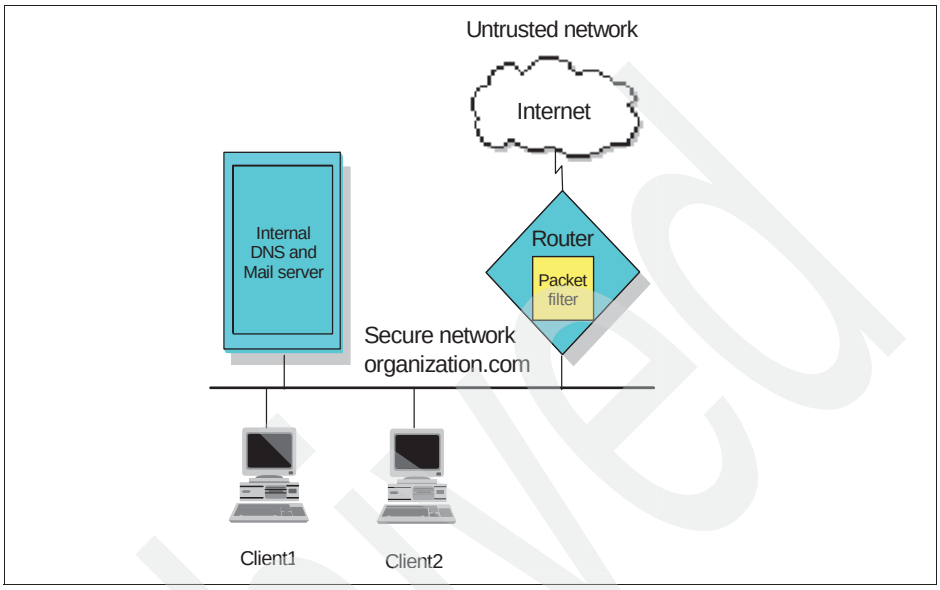


Figure 22-19 Packet-filtering firewall

The filter will allow some of the hosts on the internal network to be directly accessed from the external network. Such hosts need their own authorization mechanism and need to be updated regularly in case of any attacks.

Dual-homed gateway firewall

A dual-homed host has at least two network interfaces and therefore at least two IP addresses. IP forwarding is disabled in the firewall, thus all IP traffic between the two interfaces is broken at the firewall (see Figure 22-20 on page 807). Therefore, there is no way for a packet to pass the firewall except through the related proxy or SOCKS service. Unlike the packet-filtering firewalls, dual-homed gateway firewalls make sure that any attack that comes from an unknown service will be blocked. A dual-homed gateway implements the method in which everything not specifically permitted is denied.

Archived

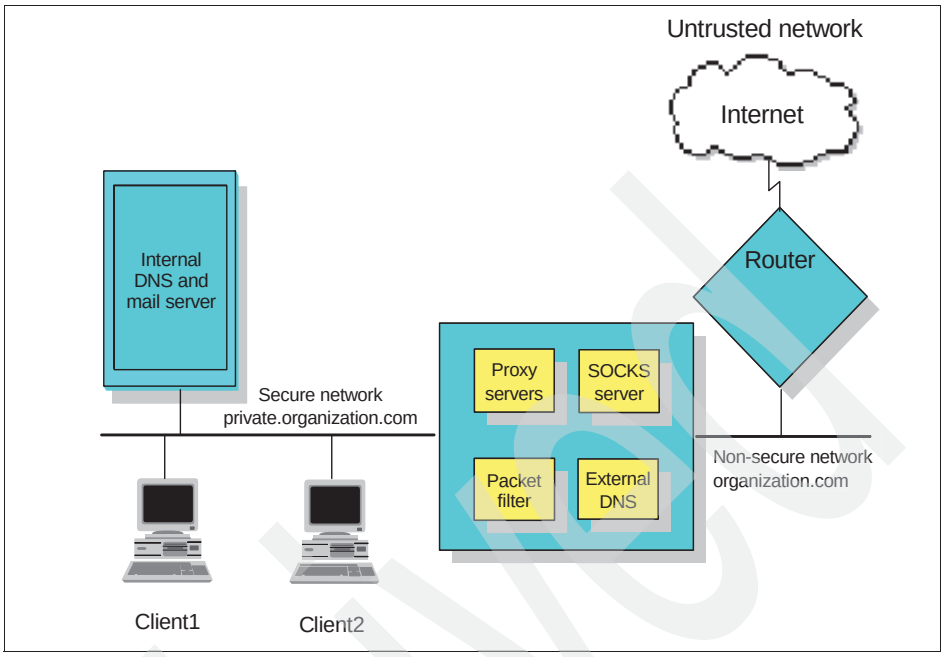


Figure 22-20 Dual-homed firewall

If an information server (such as a Web or FTP server) needs to be located to give access to both inside and outside users, it can either be installed inside the protected network or it can be installed between the firewall and the router, which is relatively insecure. If it is installed beyond the firewall, the firewall must have the related proxy services to give access to the information server from inside the secure network. If the information server is installed between the firewall and the router, the router must be capable of packet filtering and configured accordingly. This type of firewall is called a screened host firewall and discussed in the following section.

Screened host firewall

This type of firewall consists of a packet-filtering router and an application-level gateway. The host containing the application-level gateway is known as a bastion host. The router is configured to forward all untrusted traffic to the bastion host and in some cases also to the information server (see Figure 22-21 on page 808). Because the internal network is on the same subnet as the bastion host, the security policy can allow internal users to access outside networks directly or force them to use proxy services to access the outside network. This can be achieved by configuring the router filter rules so that the router only accepts outbound traffic originating from the bastion host.

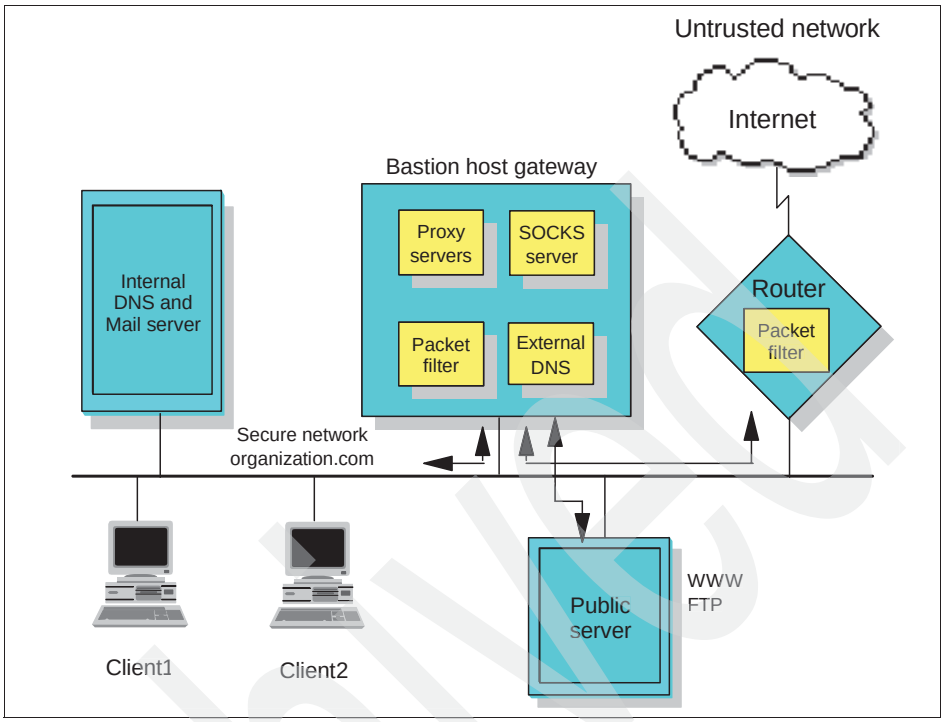


Figure 22-21 Screened host firewall

This configuration allows an information server to be placed between the router and the bastion host. Again, the security policy determines whether the information server will be accessed directly by either outside users or internal users, or if it will be accessed through the bastion host. If strong security is needed, traffic from both the internal network to the information server and from outside to the information server can go through the bastion host.

In this configuration, the bastion host can be a standard host or, if a more secure firewall system is needed, it can be a dual-homed host. In this case, all internal traffic to the information server and to the outside through the router is automatically forced to pass the proxy server on the dual-homed host. The bastion host is then the only system that can be accessed from the outside. No one should be permitted to log on to the bastion host; otherwise, an intruder might log on the system and change the configuration to bypass the firewall.

Screened subnet firewall (demilitarized zone)

This type of firewall consists of two packet-filtering routers and a bastion host. Screened subnet firewalls provide the highest level security among the different firewall types (see Figure 22-22 on page 809). This is achieved by creating a

demilitarized zone (DMZ) between the external and internal network so that the outer router only permits access from the outside to the bastion host (possibly to the information server) and the inner router only permits access from the internal network to the bastion host. The routers force all inbound and outbound traffic through the bastion host. This provides strong security because an intruder has to penetrate three separate systems to reach the internal network.

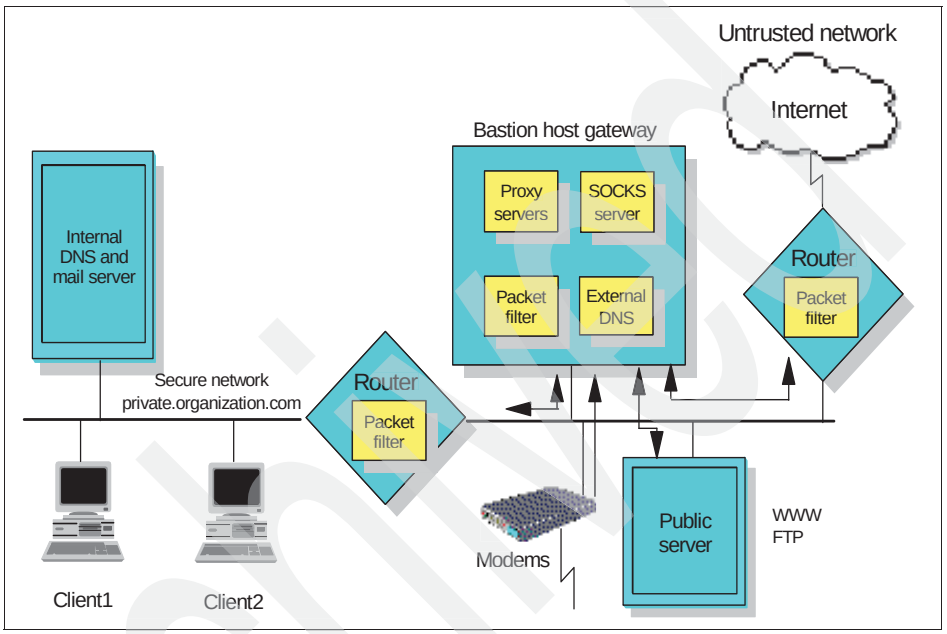


Figure 22-22 Screened subnet firewall

One of the significant benefits of the DMZ is that because the routers force the systems on both external and internal networks to use the bastion host, there is no need for the bastion host to be a dual-homed host. This provides much faster throughput than achieved by a dual-homed host. Of course, this is complicated and some security problems might be caused by improper router configurations.

22.4 IP Security Architecture (IPSec)

This section examines, in detail, the IPSec framework and its three main components, Authentication Header (AH), Encapsulated Security Payload (ESP), and Internet Key Exchange (IKE). We discuss the header formats, the specific cryptographic features, and the different modes of application.

IPSec adds integrity checking, authentication, encryption, and replay protection to IP packets. It is used for end-to-end security and also for creating secure tunnels between gateways.

IPSec was designed for interoperability. When correctly implemented, it does not affect networks and hosts that do not support it. IPSec is independent of the current cryptographic algorithms; it can accommodate new ones as they become available. It works both with IPv4 and IPv6. In fact, IPSec is a mandatory component of IPv6.

IPSec uses state-of-the-art cryptographic algorithms. The specific implementation of an algorithm for use by an IPSec protocol is often called a *transform*. For example, the DES algorithm used by ESP is called the ESP DES-CBC transform. The transforms, like the protocols, are published in the RFCs.

22.4.1 Concepts

Two major IPSec concepts need to be clarified: Security Associations and tunneling. We describe these concepts in the following sections.

Security Associations

The concept of a Security Association (SA) is fundamental to IPSec. An SA is a unidirectional (simplex) logical connection between two IPSec systems, uniquely identified by the following triple:

```
<Security Parameter Index, IP destination address, security protocol>
```

The definition of the members is as follows:

- Security parameter index (SPI)
  This is a 32-bit value used to identify different SAs with the same destination address and security protocol. The SPI is carried in the header of the security protocol (AH or ESP). The SPI has only local significance, as defined by the creator of the SA. SPI values in the range 1 to 255 are reserved by the Internet Assigned Numbers Authority (IANA). The SPI value of 0 must be used for local implementation-specific purposes only. RFC 2406 states that a value of 0 must not be transmitted. Generally, the SPI is selected by the destination system during SA establishment.
- IP destination address
  This address can be a unicast, broadcast, or multicast IP address. However, currently SA management mechanisms are defined only for unicast addresses.
- Security protocol
  This can be either AH or ESP.

An SA can be in either of two modes, transport or tunnel, depending on the mode of the protocol in that SA. You can find the explanation of these protocol modes later in this chapter.

SAs are simplex, thus, for bidirectional communication between two IPSec systems, there must be two SAs defined, one in each direction.

A single SA gives security services to the traffic carried by it either by using AH or ESP, but not both. In other words, for a connection that needs to be protected by both AH and ESP, two SAs must be defined for each direction. In this case, the set of SAs that define the connection is referred to as an *SA bundle*. The SAs in the bundle do not have to terminate at the same endpoint. For example, a mobile host can use an AH SA between itself and a firewall and a nested ESP SA that extends to a host behind the firewall.

An IPSec implementation maintains two databases related to SAs:

- Security Policy Database (SPD)
  The Security Policy Database specifies what security services are to be offered to the IP traffic, depending on factors such as source, destination, whether it is inbound, outbound, and so on. It contains an ordered list of policy entries, separate for inbound and outbound traffic. These entries might specify that some traffic must bypass the IPSec processing, some must be discarded, and the rest must be processed by the IPSec module. Entries in this database are similar to firewall rules or packet filters.
- Security Association Database (SAD)
  The Security Association Database contains parameter information about each SA, such as AH or ESP algorithms and keys, sequence numbers, protocol mode, and SA lifetime. For outbound processing, an SPD entry points to an entry in the SAD. That is, the SPD determines which SA is to be used for a given packet. For inbound processing, the SAD is consulted to determine how the packet must be processed.

Note: The user interface of an IPSec implementation usually hides or presents these databases in a friendlier way.

Tunneling

Tunneling or encapsulation is a common technique in packet-switched networks. It consists of wrapping a packet in a new one. That is, a new header is attached to the original packet. The entire original packet becomes the payload of the new one, as shown in Figure 22-23.

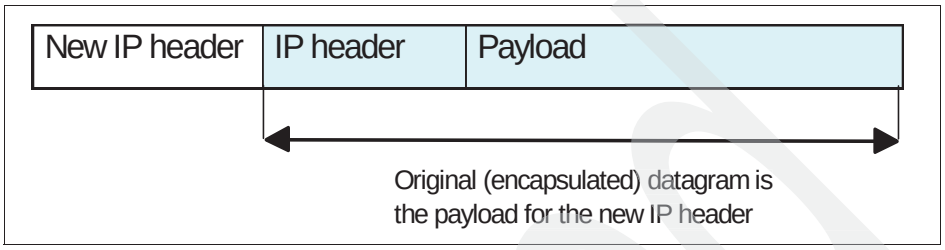


Figure 22-23 IP tunneling

In general, tunneling is used to carry traffic of one protocol over a network that does not support that protocol directly. For example, NetBIOS or IPX can be encapsulated in IP to carry it over a TCP/IP WAN link. In the case of IPSec, IP is tunneled through IP for a slightly different purpose: To provide total protection, including the header of the encapsulated packet. If the encapsulated packet is encrypted, an intruder cannot figure out, for example, the destination address of that packet. (Without tunneling, the intruder could.) The internal structure of a private network can be concealed in this way.

Tunneling requires intermediate processing of the original packet while en-route. The destination specified in the outer header, usually an IPSec firewall or router, receives the tunneled packet, extracts the original packet, and sends it to the ultimate destination. The processing cost is compensated by the extra security.

A notable advantage of IP tunneling is the possibility to exchange packets with private IP addresses between two intranets over the public Internet, which requires globally unique addresses. Because the encapsulated header is not processed by the Internet routers, only the endpoints of the tunnel (the gateways) need to have globally assigned addresses; the hosts in the intranets behind them can be assigned private addresses (for example, 10.x.x.x). Because globally unique IP addresses are becoming a scarce resource, this interconnection method gains importance.

Note: IPSec tunneling is modeled after RFC 2003 – IP Encapsulation within IP. It was originally designed for Mobile IP, an architecture that allows a mobile host to keep its home IP address even if attached to remote or foreign subnets. See 7.1, “Mobile IP overview” on page 276.

22.4.2 Authentication Header (AH)

AH is used to provide integrity and authentication to IP datagrams. Replay protection is also possible. Although its usage is optional, the replay protection service must be implemented by any IPSec-compliant system. The services are connectionless, that is, they work on a per-packet basis. AH is used in two modes, transport mode and tunnel mode.

AH authenticates as much of the IP datagram as possible. In transport mode, some fields in the IP header change en-route and their value cannot be predicted by the receiver. These fields are called *mutable* and are not protected by AH. The mutable IPv4 fields are:

- Type of service (TOS)
- Flags
- Fragment offset
- Time to live (TTL)
- Header checksum

When protection of these fields is required, tunneling must be used. The payload of the IP packet is considered immutable and is always protected by AH.

AH is identified by protocol number 51, assigned by the IANA. The protocol header (IPv4, IPv6, or extension) immediately preceding the AH contains this value in its protocol (IPv4) or Next header (IPv6, extension) field.

AH processing is applied only to non-fragmented IP packets. However, an IP packet with AH applied can be fragmented by intermediate routers. In this case, the destination first reassembles the packet and then applies AH processing to it. If an IP packet that appears to be a fragment (offset field is non-zero, or the More Fragments bit is set) is input to AH processing, it is discarded. This prevents the so-called *overlapping fragment attack*, which misuses the fragment reassembly algorithm in order to create forged packets and force them through a firewall.

Packets that fail authentication are discarded and never delivered to upper layers. This mode of operation greatly reduces the chances of successful denial-of-service attacks, which aim to block the communication of a host or gateway by flooding it with bogus packets.

Archived

AH format

The AH format is described in RFC 2402. Figure 22-24 shows the position of the Authentication Header fields in the IP packet.

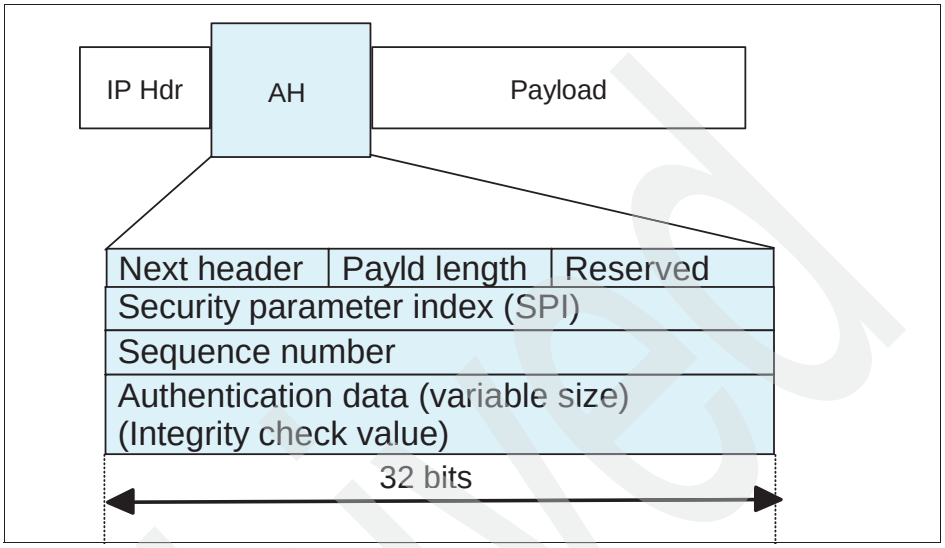


Figure 22-24 AH format

The fields are as follows:

Next header The next header *t* is an 8-bit field that identifies the type of what follows. The value of this field is chosen from the set of IP protocol numbers defined in the most recent *Assigned Numbers* RFC from the Internet Assigned Numbers Authority (IANA). In other words, the IP header protocol field is set to 51, and the value that would have gone in the protocol field goes in the AH next header field.

Payload length This field is 8 bits long and contains the length of the AH header expressed in 32-bit words, minus 2. It does not relate to the actual payload length of the IP packet as a whole. If default options are used, the value is 4 (three 32-bit fixed words plus three 32-bit words of authentication data minus two).

Reserved This field is reserved for future use. Its length is 16 bits and it is set to zero.

Security parameter index (SPI)
This field is 32 bits in length. See “Security Associations” on page 810 for a definition.

Sequence number This 32-bit field is a monotonically increasing counter, which is used for replay protection. Replay protection is optional; however, this field is mandatory. The sender always includes this field, and it is at the discretion of the receiver to process it or not. At the establishment of an SA, the sequence number is initialized to zero. The first packet transmitted using the SA has a sequence number of 1. Sequence numbers are not allowed to repeat. Therefore, the maximum number of IP packets that can be transmitted on any given SA is 2^{32}-1. After the highest sequence number is used, a new SA, and consequently a new key, are established. Anti-replay is enabled at the sender by default. If upon SA establishment the receiver chooses not to use it, the sender need not be concerned with the value in this field anymore.

Notes: Typically, the anti-replay mechanism is not used with manual key management. The original AH specification in RFC 1826 did not discuss the concept of sequence numbers. Older IPSec implementations that are based on that RFC can therefore not provide replay protection.

Authentication data This is a variable-length field containing the Integrity Check Value (ICV), and is padded to 32 bits for IPv4 or 64 bits for IPv6. The ICV for each packet is calculated with the algorithm selected at SA initialization. As its name implies, it is used by the receiver to verify the integrity of the incoming packet.

In theory, any MAC algorithm can be used to calculate the ICV. The specification requires that HMAC-MD5-96 and HMAC-SHA-1-96 must be supported. The old RFC 1826 requires Keyed MD5. In practice, Keyed SHA-1 is also used. Implementations usually support two to four algorithms.

When doing the ICV calculation, the mutable fields are considered to be filled with zero.

Ways of using AH

AH can be used in two ways: transport mode and tunnel mode.

AH in transport mode

In this mode, the authentication header is inserted immediately after the IP header, as shown in Figure 22-25. If the datagram already has IPSec headers, the AH is inserted before them.

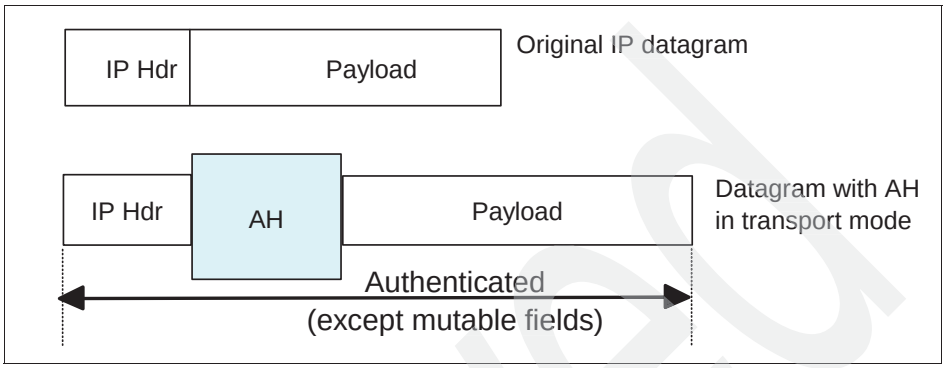


Figure 22-25 Authentication Header in transport mode

Transport mode is used by hosts, not by gateways. Gateways are not required to support transport mode.

The advantage of transport mode is fewer processing costs. The disadvantage is that mutable fields are not authenticated.

AH in tunnel mode

With this mode, the tunneling concept is applied, a new IP datagram is constructed and the original IP datagram is made the payload of it. AH in transport mode is applied to the resulting datagram. See Figure 22-26 for an illustration.

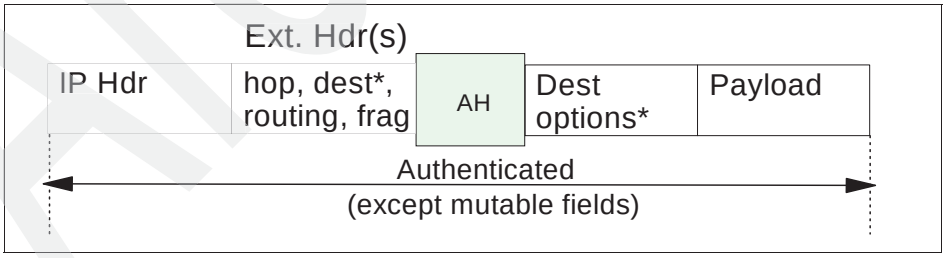


Figure 22-26 Authentication Header in tunnel mode

Tunnel mode is used whenever either end of a Security Association is a gateway. Therefore, between two firewalls, tunnel mode is always used.

Gateways often also support transport mode. This mode is allowed when the gateway acts as a host, that is, in cases when traffic is destined to the gateway itself. For example, SNMP commands can be sent to the gateway using transport mode.

In tunnel mode, the outer headers' IP addresses do not need to be the same as the inner headers' addresses. For example, two security gateways can operate an AH tunnel that is used to authenticate all traffic between the networks they connect together. This is a very typical mode of operation.

The advantages of tunnel mode include total protection of the encapsulated IP datagram and the possibility of using private addresses. However, there are extra processing costs associated with this mode.

Note: The original AH specification in RFC 1825 only mentions tunnel mode in passing, not as a requirement. Because of this, there are IPSec implementations based on that RFC that do not support AH in tunnel mode.

IPv6 considerations

AH is an integral part of IPv6 (see 9.2.1, “Extension headers” on page 333). In an IPv6 environment, AH is considered an end-to-end payload and it appears after hop-by-hop, routing, and fragmentation extension headers. The destination options extension headers can appear either before or after the Authentication Header. Figure 22-27 illustrates the positioning of AH in transport mode for a typical IPv6 packet. The position of the extension headers marked with an asterisk (*) is variable, if present at all.

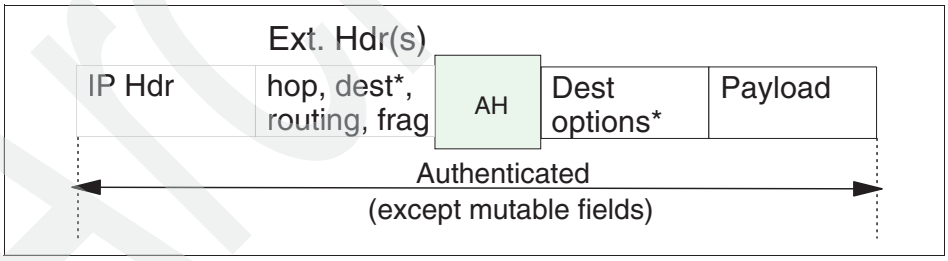


Figure 22-27 AH in transport mode for IPv6

For a detailed description of AH in IPv6, refer to RFC 2402.

22.4.3 Encapsulating Security Payload (ESP)

ESP is used to provide integrity check, authentication, and encryption to IP datagrams. Optional replay protection is also possible. These services are

connectionless, in that they operate on a per-packet basis. The set of desired services are selectable upon SA establishment. However, some restrictions apply:

- Integrity check and authentication are used together.
- Replay protection is selectable only in conjunction with integrity check and authentication.
- Replay protection can be selected only by the receiver.

Encryption can be selected independently of other services. It is highly recommended that, if encryption is enabled, integrity check and authentication be turned on. If only encryption is used, intruders can forge packets in order to mount cryptanalytic attacks.

Although both authentication (with integrity check) and encryption are optional, at least one of them is always selected; otherwise, you would not be using ESP.

ESP is identified by protocol number 50, as assigned by the IANA. The protocol header (IPv4, IPv6, or extension) immediately preceding the AH header will contain this value in its protocol (IPv4) or the next header field (IPv6, extension).

ESP processing is applied only to non-fragmented IP packets. However, an IP packet with ESP applied can be fragmented by intermediate routers. In this case, the destination first reassembles the packet and then applies ESP processing to it. If an IP packet that appears to be a fragment is input to ESP processing (offset field is non-zero, or the More Fragments bit is set), it is discarded. This prevents the overlapping fragment attack mentioned in 22.4.2, “Authentication Header (AH)” on page 813.

If both encryption and authentication with integrity check are selected, the receiver first authenticates the packet and, only if this step was successful, proceeds with decryption. This mode of operation saves computing resources and reduces the vulnerability to denial-of-service attacks.

ESP packet format

The current ESP packet format is described in RFC 2406. It contains important modifications compared to the previous ESP specification, RFC 1827. The information in this section is based on RFC 2406.

The format of the ESP packet is more complicated than that of the AH packet. There is not only an ESP header, but also an ESP trailer and ESP authentication data (see Figure 22-28 on page 819). The payload is located (encapsulated) between the header and the trailer, thus the name of the protocol.

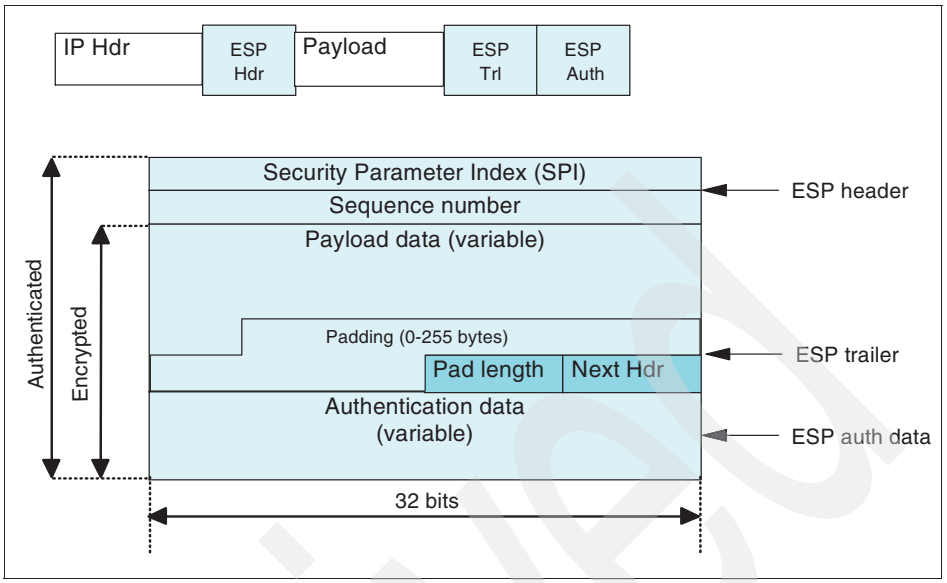


Figure 22-28 ESP header and trailer

The following fields are part of an ESP packet:

Security Parameter Index (SPI)
This field is 32 bits in length. See “AH format” on page 814 for the definition.

Sequence number This 32-bit field is a monotonically increasing counter. See “AH format” on page 814 for the definition.

Notes: Typically, the anti-replay mechanism is not used with manual key management. The original ESP specification in RFC 1827 did not discuss the concept of sequence numbers. Older IPSec implementations that are based on that RFC can therefore not provide replay protection.

Payload data The payload data field is mandatory. It consists of a variable number of bytes of data described by the next header field. This field is encrypted with the cryptographic algorithm selected during SA establishment. If the algorithm requires initialization vectors, these are also included here.
The ESP specification requires support for the DES algorithm in CBC mode (DES-CBC transform). Often, other encryption algorithms are also supported, such as triple-DES and CDMF, in the case of IBM products.

Padding Most encryption algorithms require that the input data must be an integral number of blocks. Also, the resulting ciphertext (including the padding, pad length, and next header fields) must terminate on a 4-byte boundary, so the next header field is right-aligned. For this reason, padding is included. It can also be used to hide the length of the original messages. However, this might adversely impact the effective bandwidth. Padding is an optional field (but needed for some algorithms).

Note: The encryption covers the payload data, padding, pad length and next header fields.

Pad length This 8-bit field contains the number of the preceding padding bytes. It is always present, and the value of 0 indicates no padding.

Next header The next header is an 8-bit mandatory field that shows the data type carried in the payload, for example, an upper-level protocol identifier such as TCP. The values are chosen from the set of IP protocol numbers defined by the IANA.

Authentication data This field is variable in length and contains the ICV calculated for the ESP packet from the SPI to the next header field inclusive. The authentication data field is optional. It is included only when integrity check and authentication have been selected at SA initialization time.
The ESP specifications require two authentication algorithms to be supported: HMAC with MD5 and HMAC with SHA-1. Often the simpler keyed versions are also supported by IPSec implementations.

Notes: The IP header is not covered by the ICV. The original ESP specification in RFC 1827 discusses the concept of authentication within ESP in conjunction with the encryption transform. That is, there is no authentication data field and it is left to the encryption transforms to eventually provide authentication.

Ways of using ESP

Like AH, ESP can be used in two ways: transport mode and tunnel mode.

ESP in transport mode

In this mode, the ESP header is inserted right after the IP header, as shown in Figure 22-29. If the datagram already has IPSec header or headers, the ESP header is inserted before any of those. The ESP trailer and the optional authentication data are appended to the payload.

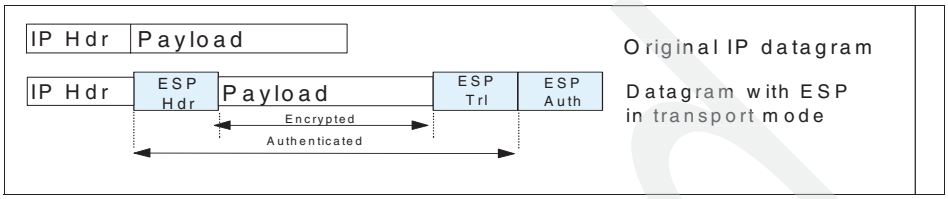


Figure 22-29 ESP in transport mode

ESP in transport mode provides neither authentication nor encryption for the IP header. This is a disadvantage, because false packets might be delivered for ESP processing. The advantage of transport mode is the lower processing cost.

As in the case of AH, ESP in transport mode is used by hosts, not gateways. Gateways are not required to support transport mode.

ESP in tunnel mode

As expected, this mode applies the tunneling principle. A new IP packet is constructed with a new IP header. ESP is then applied, as in transport mode. This is illustrated in Figure 22-30. Because the original datagram becomes the payload data for the new ESP packet, it is completely protected if both encryption and authentication are selected. However, the new IP header is still not protected.

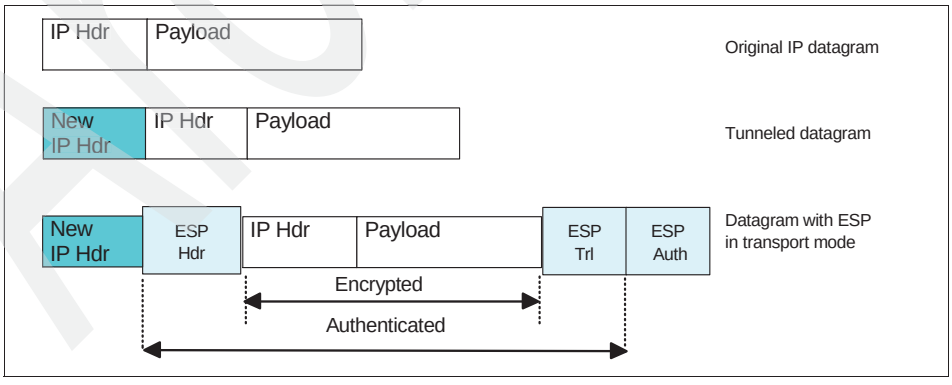


Figure 22-30 ESP in tunnel mode

Archive

The tunnel mode is used whenever either end of a Security Association is a gateway. Therefore, between two firewalls the tunnel mode is always used.

Gateways often also support transport mode. This mode is allowed when the gateway acts as a host, that is, in cases when traffic is destined to the gateway itself. For example, SNMP commands can be sent to the gateway using transport mode.

In tunnel mode the outer header's IP addresses does not need to be the same as the inner headers' addresses. For example, two security gateways can operate an ESP tunnel that is used to secure all traffic between the networks they connect together. Hosts are not required to support tunnel mode.

The advantages of tunnel mode are total protection of the encapsulated IP datagram and the possibility of using private addresses. However, there is an extra processing charge associated with this mode.

IPv6 considerations

As with AH, ESP is an integral part of IPv6 (see 9.2.1, “Extension headers” on page 333). In an IPv6 environment, ESP is considered an end-to-end payload and it appears after hop-by-hop, routing, and fragmentation extension headers. The destination options extension header(s) could appear either before or after the AH header. Figure 22-31 illustrates the positioning of the AH header in transport mode for a typical IPv6 packet. The position of the extension headers marked with an asterisk (*) is variable, if present at all.

For more details, refer to RFC 2406.

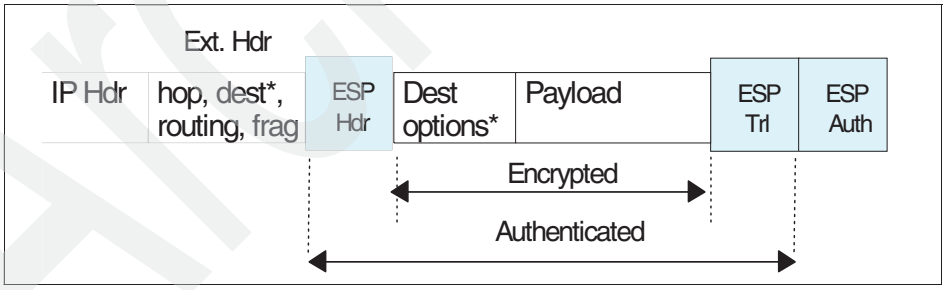


Figure 22-31 ESP in transport mode for IPv6

Two authentication protocols

Knowing about the security services of ESP, you might ask if there is really a requirement for AH. Why does ESP authentication not cover the IP header as

well? There is no official answer to these questions, but here are some points that justify the existence of two different IPSec authentication protocols:

- ESP requires strong cryptographic algorithms to be implemented, whether it will actually be used or not. There are restrictive regulations on strong cryptography in some countries. It might be troublesome to deploy ESP-based solutions in such areas. However, authentication is not regulated and AH can be used freely around the world.
- Often, only authentication is needed. AH is more performant compared to ESP with authentication only, because of the simpler format and lower processing costs. It makes sense to use AH in these cases.
- Having two different protocols means finer-grade control over an IPSec network and more flexible security options. By nesting AH and ESP, for example, you can implement IPSec tunnels that combine the strengths of both protocols.

22.4.4 Combining IPSec protocols

The AH and ESP protocols can be applied alone or in combination. Given the two modes of each protocol, there is quite a number of possible combinations. To make things more complicated, the AH and ESP SAs do not need to have identical endpoints. Luckily, out of the many possibilities, only a few make sense in real-world scenarios.

Note: RFC 2406 describes mandatory combinations that must be supported by each IPSec implementation. Other combinations may also be supported, but this might impact interoperability.

We mentioned in “Security Associations” on page 810 that the combinations of IPSec protocols are realized with SA bundles.

There are two approaches to creating an SA bundle:

- Transport adjacency: Both security protocols are applied in transport mode to the same IP datagram. This method is practical for only one level of combination.
- Iterated (nested) tunneling: The security protocols are applied in tunnel mode, in sequence. After each application, a new IP datagram is created and the next protocol is applied to it. This method has no limit in the nesting levels. However, more than three levels are impractical.

These approaches can be combined. For example, an IP packet with transport adjacency IPSec headers can be sent through nested tunnels.

When designing a VPN, limit the number of IPSec processing stages. In our view, three stages is the limit beyond which further processing has no benefits. Two stages are sufficient for almost all cases.

Note that, in order to be able to create an SA bundle in which the SAs have different endpoints, at least one level of tunneling must be applied. Transport adjacency does not allow for multiple source/destination addresses, because only one IP header is present.

The practical principle of the combined usage is that, upon the receipt of a packet with both protocol headers, the IPSec processing sequence should be authentication followed by decryption. It is common sense not to bother with decryption of packets of uncertain origin.

Following this principle, the sender first applies ESP and then AH to the outbound traffic. In fact, this sequence is an explicit requirement for transport mode IPSec processing. When using both ESP and AH, a new question arises: Should ESP authentication be turned on? AH authenticates the packet anyway. The answer is simple.

Turning on ESP authentication makes sense only when the ESP SA extends beyond the AH SA. For example, ESP can be used end-to-end, while AH only goes as far as the remote gateway. In this case, not only does it make sense to use ESP authentication, but we highly recommend doing so to avoid spoofing attacks within the intranet.

As far as the modes are concerned, transport mode is usually used between the endpoints of a connection and tunnel mode is usually used between two machines when at least one of them is a gateway.

Let us take a look at the different ways of using the IPSec protocols, from the simplest to the more complicated nested setups.

Archive

Case 1: End-to-end security

As shown in Figure 22-32, two hosts are connected through the Internet (or an intranet) without any IPSec gateway between them. They can use ESP, AH, or both. Either transport or tunnel mode can be applied.

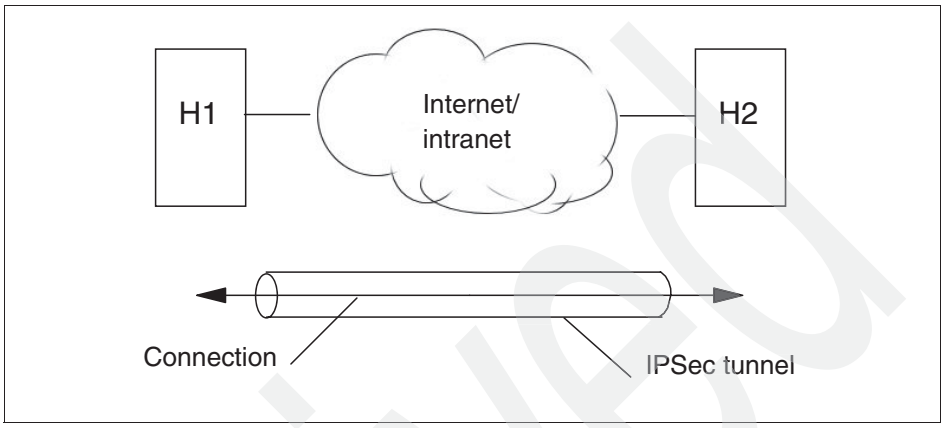


Figure 22-32 End-to-end security

The following combinations are required to be supported by any IPSec implementation:

- Transport mode
  - AH alone
  - ESP alone
  - AH applied after ESP (transport adjacency)
- Tunnel mode
  - AH alone
  - ESP alone

Case 2: Basic VPN support

We describe virtual private networks (VPNs) in 22.10, “Virtual private networks (VPNs) overview” on page 861.

Archived

Figure 22-33 illustrates the simplest IPSec VPN. The gateways G1 and G2 run the IPSec protocol stack. The hosts in the intranets are not required to support IPSec.

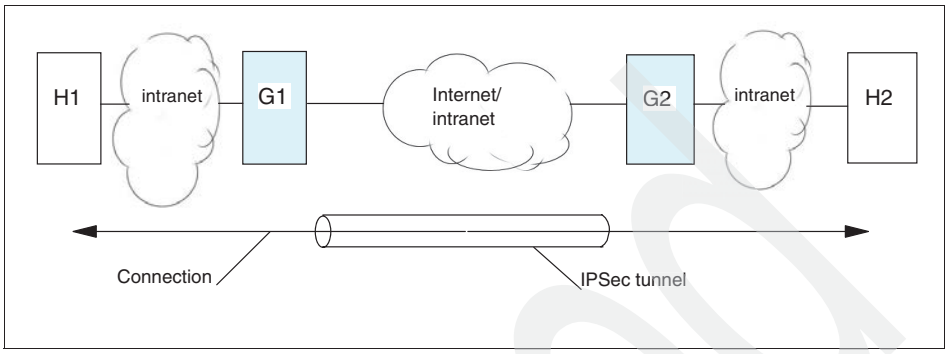


Figure 22-33 Basic VPN support

In this case, the gateways are required to support only tunnel mode, either with AH or ESP.

Combined tunnels between gateways

Although gateways are required to support either an AH tunnel or ESP tunnel, it is often desirable to have tunnels between gateways that combine the features of both IPSec protocols.

The IBM IPSec implementations support this type of combined AH-ESP tunnels. The order of the headers is user selectable by setting the tunnel policy.

A combined tunnel between gateways does not mean that iterated tunneling takes place. Because the SA bundles comprising the tunnel have identical endpoints, it is inefficient to do iterated tunneling. Instead, one IPSec protocol is applied in tunnel mode and the other in transport mode, which can be conceptually thought of as a combined AH-ESP tunnel. An equivalent approach is to IP tunnel the original datagram and then apply transport adjacency IPSec processing to it. The result is that we have an outer IP header followed by the IPSec headers in the order set by the tunnel policy, and then the original IP packet, as shown in Figure 22-34 on page 827. This is the packet format in a combined AH-ESP tunnel between two IBM firewalls.

Note: ESP authentication data was not present in early implementations of the IBM firewall.

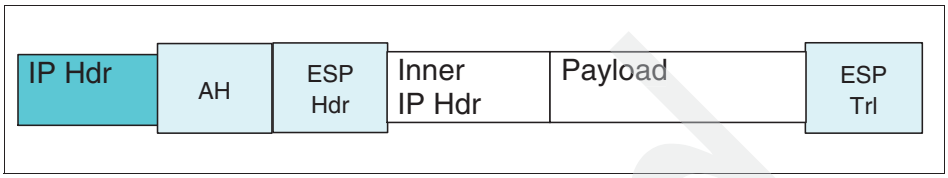


Figure 22-34 Combined AH-ESP tunnel

Case 3: End-to-end security with VPN support

This case is a combination of cases 1 and 2 and does not raise new IPSec requirements for the machines involved (see Figure 22-35). The big difference from case 2 is that now the hosts are also required to support IPSec.

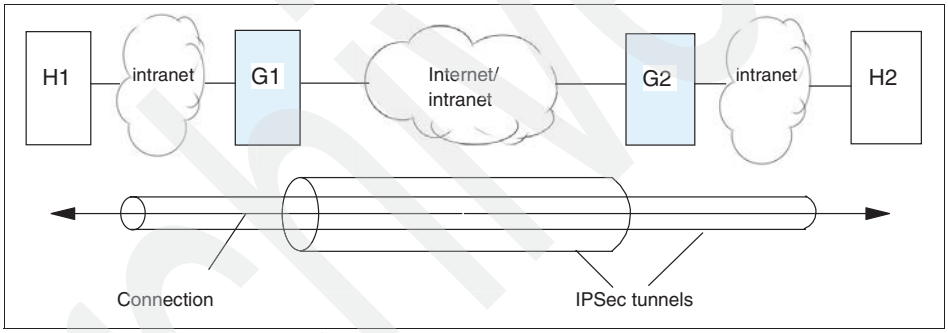


Figure 22-35 End-to-end security with VPN support

In a typical setup, the gateways use AH in tunnel mode, while the hosts use ESP in transport mode. An enhanced security version might use a combined AH-ESP tunnel between the gateways. In this way, the ultimate destination addresses are encrypted; the whole packet traveling the Internet would be authenticated and the carried data double encrypted. This is the only case when three stages of IPSec processing might be useful, however, at a cost—the performance impact is considerable.

AH tunneling of ESP transport

Let us look in more detail at the common combination of using AH tunneling to protect ESP traffic in transport mode.

Figure 22-36 shows in detail how this combination is realized. Consider that host H1 in Figure 22-35 on page 827 sends an IP packet to host H2. Here is what happens:

1. Host H1 constructs the IP packet and applies ESP transport to it. H1 then sends the datagram to gateway G1, the destination address being H2.
2. Gateway G1 realizes that this packet should be routed to G2. Upon consulting its IPSec databases (SPD and SAD), G1 concludes that AH in tunnel mode must be applied before sending the packet out. It does the required encapsulation. Now the IP packet has the address of G2 as its destination, the ultimate destination H2 being encapsulated.
3. Gateway G2 receives the AH-tunneled packet. It is destined to itself, so it authenticates the datagram and strips off the outer header. G2 sees that the payload is yet another IP packet (that one sent by H1) with destination H2, so it forwards to H2. G2 does not care that this packet has an ESP header.
4. Finally H2 receives the packet. Because this is the destination, ESP-transport processing is applied and the original payload retrieved.

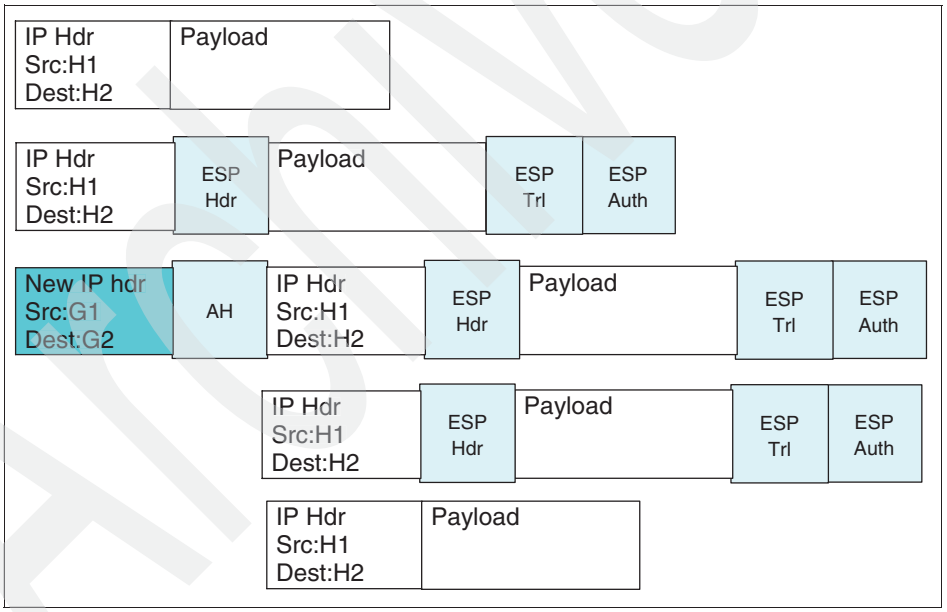


Figure 22-36 Nesting of IPSec protocols

Archive

Case 4: Remote access

This case, shown in Figure 22-37, applies to remote hosts that use the Internet to reach a server in the organization protected by a firewall. The remote host typically uses a PPP dial-in connection to an ISP.

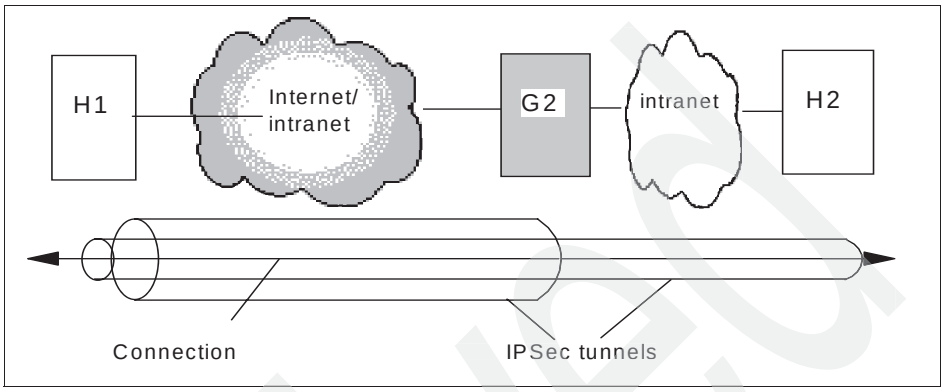


Figure 22-37 Remote access

Between the remote host H1 and the firewall G2, only tunnel mode is required. The choices are the same as in case 2. Between the hosts themselves, either tunnel mode or transport mode can be used, with the same choices as in case 1.

A typical setup is to use AH in tunnel mode between H1 and G2 and ESP in transport mode between H1 and H2. Older IPSec implementations that do not support AH in tunnel mode cannot implement this.

It is also common to create a combined AH-ESP tunnel between the remote host H1 and the gateway G2. In this case, H1 can access the whole intranet using just one SA bundle, while if it were using the setup shown in Figure 22-37, it only could access one host with one SA bundle.

22.4.5 Internet Key Exchange (IKE) protocol

The Internet Key Exchange (IKE) framework, previously referred to as ISAKMP/Oakley, supports automated negotiation of Security Associations, and automated generation and refresh of cryptographic keys. The ability to perform these functions with little or no manual configuration of machines is a critical element to any enterprise-scale IPSec deployment.

Archived

Before describing the details of the key exchange and update messages, some explanations are necessary:

- Internet Security Association and Key Management Protocol (ISAKMP)

  A framework that defines the management of Security Associations (negotiate, modify, delete) and keys, and it also defines the payloads for exchanging key generation and authentication data. ISAKMP itself does not define any key exchange protocols, and the framework it provides can be applied to security mechanisms in the network, transport, or application layer, and also to itself.

- Oakley

  A key exchange protocol that can be used with the ISAKMP framework to exchange and update keying material for Security Associations.

- Domain of Interpretation (DOI)

  Definition of a set of protocols to be used with the ISAKMP framework for a particular environment; also a set of common definitions shared with those protocols regarding the syntax of SA attributes and payload contents, namespace of cryptographic transforms, and so on. In relation to IPSec, the DOI instantiates ISAKMP for use with IP.

- Internet Key Exchange (IKE)

  A protocol that uses parts of ISAKMP and parts of the Oakley and SKEME key exchange protocols to provide management of keys and Security Associations for the IPSec AH and ESP protocols and for ISAKMP itself.

Protocol overview

ISAKMP requires that all information exchanges must be both encrypted and authenticated, so that no one can eavesdrop on the keying material. The keying material will be exchanged only among authenticated parties. This is required because the ISAKMP procedures deal with initializing the keys, so they must be capable of running over links where no security can be assumed to exist.

In addition, the ISAKMP methods have been designed with the explicit goals of providing protection against several well-known exposures:

- Denial of service: The messages are constructed with unique *cookies* that can be used to quickly identify and reject invalid messages without the need to execute processor-intensive cryptographic operations.
- Man-in-the-middle: Protection is provided against the common attacks such as deletion of messages, modification of messages, reflecting messages back to the sender, replaying of old messages, and redirection of messages to unintended recipients.

- Perfect Forward Secrecy (PFS): Compromise of past keys provides no useful clues for breaking any other key, whether it occurred before or after the compromised key. That is, each refreshed key will be derived without any dependence on predecessor keys.

The following authentication methods are defined for IKE:

- Pre-shared key
- Digital signatures (DSS and RSA)
- Public key encryption (RSA and revised RSA)

The robustness of any cryptography-based solution depends much more strongly on keeping the keys secret than it does on the actual details of the chosen cryptographic algorithms. Therefore, the IETF IPSec Working Group has prescribed a set of extremely robust Oakley exchange protocols. It uses a two-phase approach.

Phase 1

This set of negotiations establishes a master secret from which all cryptographic keys will be derived for protecting the users' data traffic. In the most general case, public key cryptography is used to establish an ISAKMP Security Association between systems and to establish the keys that will be used to protect the ISAKMP messages that will flow in the subsequent phase 2 negotiations. Phase 1 is concerned only with establishing the protection suite for the ISAKMP messages themselves, but it does not establish any Security Associations or keys for protecting user data.

In phase 1, the cryptographic operations are the most processor-intensive, but need only be done infrequently, and a single phase 1 exchange can be used to support multiple subsequent phase 2 exchanges. As a guideline, phase 1 negotiations are executed once a day or maybe once a week, while phase 2 negotiations are executed once every few minutes.

Phase 2

Phase 2 exchanges are less complex, because they are used only after the security protection suite negotiated in phase 1 has been activated. A set of communicating systems negotiate the Security Associations and keys that will protect user data exchanges. Phase 2 ISAKMP messages are protected by the ISAKMP Security Association generated in phase 1. Phase 2 negotiations generally occur more frequently than phase 1. For example, a typical application of a phase 2 negotiation is to refresh the cryptographic keys once every two to three minutes.

Permanent identifiers

The IKE protocol also offers a solution even when the remote host's IP address is not known in advance. ISAKMP allows a remote host to identify itself by a *permanent* identifier, such as a name or an e-mail address. The ISAKMP phase 1 exchanges will then authenticate the remote host's permanent identity using public key cryptography:

- Certificates create a binding between the permanent identifier and a public key. Therefore, ISAKMP's certificate-based phase 1 message exchanges can authenticate the remote host's permanent identify.
- Because the ISAKMP messages themselves are carried within IP datagrams, the ISAKMP partner (for example, a firewall or destination host) can associate the remote host's dynamic IP address with its authenticated permanent identity.

Initializing Security Associations with IKE

This section outlines how ISAKMP/Oakley protocols initially establish Security Associations and exchange keys between two systems that want to communicate securely.

In the remainder of this section, the parties involved are named Host-A and Host-B. Host-A is the initiator of the ISAKMP phase 1 exchanges, and Host-B is the responder. If needed for clarity, subscripts A or B are used to identify the source of various fields in the message exchanges.

IKE phase 1: Setting up ISAKMP Security Associations

The Security Associations that protect the ISAKMP messages themselves are set up during the phase 1 exchanges. Because we are starting “cold” (no previous keys or SAs have been negotiated between Host-A and Host-B), the phase 1 exchanges use the ISAKMP Identity Protect Exchange (also known as Oakley Main Mode). Six messages are needed to complete the exchange:

- Messages 1 and 2 negotiate the characteristics of the Security Associations. Messages 1 and 2 flow in the clear for the initial phase 1 exchange, and they are unauthenticated.
- Messages 3 and 4 exchange nonces (random values) and also execute a Diffie-Hellman exchange to establish a master key (SKEYID). Messages 3 and 4 flow in the clear for the initial phase 1 exchange, and they are unauthenticated.
- Messages 5 and 6 exchange the required information for mutually authenticating the parties' identities. The payloads of messages 5 and 6 are protected by the encryption algorithm and keying material established with messages 1 through 4.

A detailed description of the phase 1 messages and exchanged information follows.

IKE phase 1, message 1

Because Host-A is the initiating party, it constructs a cleartext ISAKMP message (message 1) and sends it to Host-B. The ISAKMP message itself is carried as the payload of a UDP packet, which in turn is carried as the payload of a normal IP datagram (see Figure 22-38).

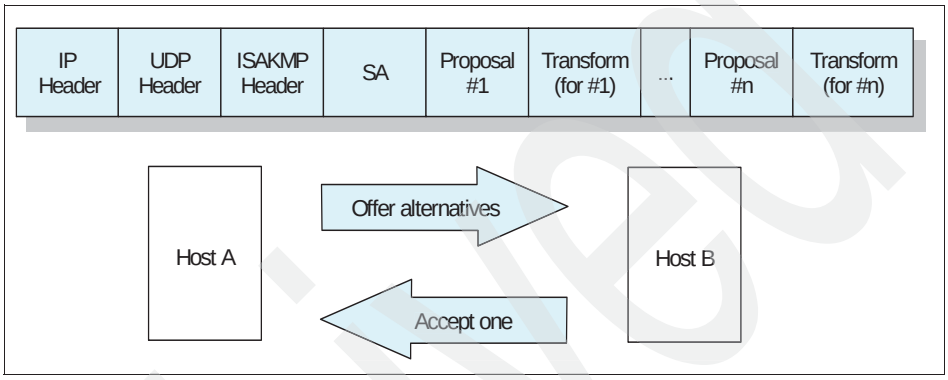


Figure 22-38 Message 1 of an ISAKMP phase 1 exchange

The source and destination addresses to be placed in the IP header are those of Host-A (initiator) and Host-B (responder), respectively. The UDP header will identify that the destination port is 500, which has been assigned for use by the ISAKMP protocol. The payload of the UDP packet carries the ISAKMP message itself.

In message 1, Host-A, the initiator, proposes a set of one or more protection suites for consideration by Host-B, the responder. Therefore, the ISAKMP message contains at least the following fields in its payload:

ISAKMP header The ISAKMP header in message 1 indicates an exchange type of Main Mode and contains a Message ID of 0. Host-A sets the Responder Cookie field to 0 and fills in a random value of its choice for the Initiator Cookie, denoted as Cookie-A.

Security Association The Security Association field identifies the Domain of Interpretation (DOI). Because the hosts plan to run IPSec protocols between themselves, the DOI is simply IP.

Proposal Payload Host-A's Proposal Payload specifies the protocol PROTO_ISAKMP and sets the SPI value to 0.

Note: For ISAKMP phase 1 messages, the SPI field within the Proposal Payload is not used to identify the ISAKMP Security Association. During phase 1, the ISAKMP SA is identified instead by the pair of values <Initiator Cookie, Responder Cookie>, both of which must be non-zero values. Because the Responder Cookie has not yet been generated by Host-B, the ISAKMP SA is not yet unambiguously identified.

Transform Payload The Transform Payload specifies KEY_OAKLEY. For the KEY_OAKLEY transform, Host-A must also specify the relevant attributes: namely, the authentication method to be used, the pseudo-random function to be used, and the encryption algorithm to be used.

Note: Multiple proposals can be included in message 1.

IKE phase 1, message 2

In message 1, Host-A proposed one or more candidate protection suites to be used to protect the ISAKMP exchanges. Host-B uses message 2 to indicate which one, if any, it will support. If Host-A proposed just a single option, Host-B merely needs to acknowledge that the proposal is acceptable.

The source and destination addresses to be placed in the IP header are those of Host-B (responder) and Host-A (initiator), respectively. The UDP header identifies that the destination port is 500, which has been assigned for use by the ISAKMP protocol. The payload of the UDP packet carries the ISAKMP message itself.

The message contents are as follows:

ISAKMP header The ISAKMP header in message 2 indicates an exchange type of Main Mode and contains a Message ID of 0. Host-B sets the Responder Cookie field to a random value, which we call Cookie-B, and copies into the Initiator Cookie field the value that was received in the Cookie-A field of message 1. The value pair <Cookie-A, Cookie-B> serves as the SPI for the ISAKMP Security Association.

Security Association The Security Association field identifies the Domain of Interpretation (DOI). Because the hosts plan to run IPSec protocols between themselves, the DOI is simply IP.

Proposal Payload Host-B's Proposal Payload specifies the protocol PROTO_ISAKMP and sets the SPI value to 0.

Transform Payload The Transform Payload specifies KEY_OAKLEY. For the KEY_OAKLEY transform, the attributes that were accepted from the proposal offered by Host-A are copied into the appropriate fields.

At this point, the properties of the ISAKMP Security Association have been agreed to by Host-A and Host-B. The identity of the ISAKMP SA has been set equal to the pair <Cookie-A, Cookie-B>. However, the identities of the parties claiming to be Host-A and Host-B have not yet been authoritatively verified.

IKE phase 1, message 3

The third message of the phase 1 ISAKMP exchange begins the exchange of the information from which the cryptographic keys will eventually be derived (see Figure 22-39 on page 836).

The ISAKMP payload exchanges two types of information.

Important: None of the messages themselves carry the actual cryptographic keys. Instead, they carry inputs that will be used by Host-A and Host-B to derive the keys locally.

Diffie-Hellman public value The Diffie-Hellman public value g^x from the initiator. The exponent x in the public value is the private value that must be kept secret.

Nonce The nonce N_i from the initiator. (Nonce is a name for a value that is considered to be random according to some very strict mathematical guidelines.)

ID If the RSA public key is used for authentication, the nonces are encrypted with the public key of the other party. Likewise for the IDs of either party, which are then also exchanged at this stage.

If authentication with revised RSA public key is used, the KE and ID payloads are encrypted with a secret key that is derived from the nonces and the encryption algorithm agreed to in messages 1 and 2, thus avoiding one CPU-intensive public key operation.

Archived

Certificates can optionally be exchanged in either case of public key authentication, as well as a hash value thereof.

These values are carried in the Key Exchange, and the Nonce and the ID fields, respectively.

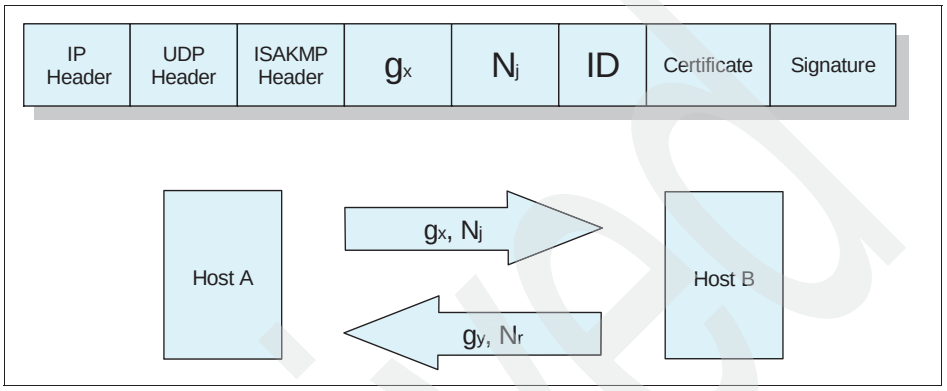


Figure 22-39 Message 3 of an ISAKMP phase 1 exchange

IKE phase 1, message 4

After receiving a Diffie-Hellman public value and a nonce from Host-A, Host-B responds by sending to Host-A its own Diffie-Hellman public value (g^y from the responder) and its nonce (N_r from the responder).

Generating the keys (phase 1)

At this point, each host knows the values of the two nonces (N_i and N_r). Each host also knows its own private Diffie-Hellman value (x and y) and also knows its partner's public value (g^x or g^y). Therefore, each side can construct the composite value g^{xy}. And finally, each side knows the values of the initiator cookie and the responder cookie.

Given all these bits of information, each side can then independently compute identical values for the following quantities:

- SKEYID: This collection of bits is sometimes referred to as keying material, because it provides the raw input from which actual cryptographic keys will be derived later in the process. It is obtained by applying the agreed-to keyed pseudorandom function (prf) to the known inputs:
  - For digital signature authentication:

    $SKEYID = prf(N_i, N_r, g^{xy})$

– For authentication with public keys:

SKEYID = prf(hash(N_i, N_r), CookieA, CookieB)

– For authentication with a pre-shared key:

SKEYID = prf(pre-shared key, N_i, N_r)

- Having computed the value SKEYID, each side then proceeds to generate two cryptographic keys and some additional keying material:
  – SKEYID_d is keying material that will be subsequently used in phase 2 to derive the keys that will be used in non-ISAKMP SAs for protecting user traffic:

    SKEYID_d = prf(SKEYID, g^{xy}, CookieA, CookieB, 0)

  – SKEYID_a is the key used for authenticating ISAKMP messages:

    SKEYID_a = prf(SKEYID, SKEYID_d, g^{xy}, CookieA, CookieB, 1)

  – SKEYID_e is the key used for encrypting ISAKMP exchanges:

    SKEYID_e = prf(SKEYID, SKEYID_a, g^{xy}, CookieA, CookieB, 2)

At this point in the protocol, both Host-A and Host-B have derived identical authentication and encryption keys that they will use to protect the ISAKMP exchanges. And they have also derived identical keying material from which they will derive keys to protect user data during phase 2 of the ISAKMP negotiations. However, at this point, the two parties' identities still have not been authenticated to one another.

IKE phase 1, message 5

At this point in the phase 1 flows, the two hosts exchange identity information with each other to authenticate themselves. As shown in Figure 22-40, the ISAKMP message carries an identity payload, a signature payload, and an optional certificate payload. Host-A uses message 5 to send information to Host-B that will allow Host-B to authenticate Host-A.

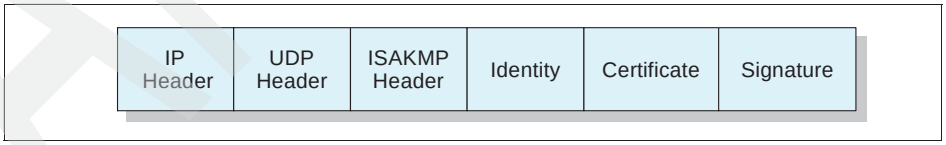


Figure 22-40 Message 5 of an ISAKMP phase 1 exchange

When an actual certificate is present in the Certificate Payload field, the receiver can use the information directly, after verifying that it has been signed with a valid signature of a trusted certificate authority. If there is no certificate in the message, it is the responsibility of the receiver to obtain a certificate using some implementation method. For example, it can send a query to a trusted certificate

authority using a protocol such as LDAP, or it can query a secure DNS server, or it can maintain a secure local cache that maps previously used certificates to their respective ID values, or it can send an ISAKMP Certificate Request message to its peer, who must then immediately send its certificate to the requester.

Note: The method for obtaining a certificate is a local option, and is not defined as part of IKE. In particular, it is a local responsibility of the receiver to check that the certificate in question is still valid and has not been revoked.

There are several points to bear in mind:

- At this stage of the process, all ISAKMP payloads, whether in phase 1 or phase 2, are encrypted, using the encryption algorithm (negotiated in messages 1 and 2) and the keys (derived from the information in messages 3 and 4). The ISAKMP header itself, however, is still transmitted in the clear.
- In phase 1, IPSec's ESP protocol is not used; that is, there is no ESP header. The recipient uses the encryption bit in the Flags field of the ISAKMP header to determine if encryption has been applied to the message. The pair of values <CookieA, CookieB>, which serve as an SPI for phase 1 exchanges, provide a pointer to the correct algorithm and key to be used to decrypt the message.
- The digital signature, if used, is not applied to the ISAKMP message itself. Instead, it is applied to a hash of information that is available to both Host-A and Host-B.
- The identity carried in the identity payload does not necessarily bear any relationship to the source IP address; however, the identity carried in the identity payload must be the identity to which the certificate, if used, applies.

Host-A (the initiator) generates the following hash function, and then places the result in the Signature Payload field:

$$HASH\_I = prf(SKEYID, g^x, g^y, CookieA, CookieB, SA_p, ID_A)$$

If digital signatures were used for authentication, this hash will also be signed by Host-A.

ID_A is Host-A's identity information that was transmitted in the identity payload of this message, and SA_p is the entire body of the SA payload that was sent by Host-A in message 1, including all proposals and all transforms proposed by Host-A. The cookies, public Diffie-Hellman values, and SKEYID were explicitly carried in messages 1 through 4, or were derived from their contents.

IKE phase 1, message 6

After receiving message 5 from Host-A, Host-B verifies the identity of Host-A by validating the hash.

If digital signatures were used for authentication, the signature of this hash are verified by Host-B.

If this is successful, Host-B sends message 6 to Host-A to allow Host-A to verify the identity of Host-B.

The structure of message 6 is the same as that of message 5, with the obvious changes that the identity payload and the certificate payload now pertain to Host-B:

$$HASH\_R = prf(SKEYID, g^y, g^x, CookieB, CookieA, SA_p, ID_B)$$

Notice that the order in which Diffie-Hellman public values and the cookies appear has been changed, and the final term now is the identity payload that Host-B has included in message 6.

If digital signatures were used for authentication, this hash is also signed by Host-B, which is different from the one previously signed by Host-A.

When Host-A receives message 6 and verifies the hash or digital signature, the phase 1 exchanges are then complete. At this point, each participant has authenticated itself to its peer. Both have agreed on the characteristics of the ISAKMP Security Associations, and both have derived the same set of keys (or keying material).

Miscellaneous phase 1 facts

There are several miscellaneous facts worth noting:

- Regardless of the specific authentication mechanism that is used, there will be six messages exchanged for the Oakley Main Mode. However, the content of the individual messages differs, depending on the authentication method.
- Although Oakley exchanges make use of both encryption and authentication, they do not use either IPSec's ESP or AH protocol. ISAKMP exchanges are protected with application-layer security mechanisms, not with network-layer security mechanisms.
- ISAKMP messages are sent using UDP. There is no guaranteed delivery for them.
- The only way to identify that an ISAKMP message is part of a phase 1 flow rather than a phase 2 flow is to check the Message ID field in the ISAKMP header. For phase 1 flows, it must be 0, and (although not explicitly stated in the ISAKMP documents) for phase 2 flows, it must be non-zero.

IKE phase 2: Setting up protocol Security Associations

After completing the phase 1 negotiation process to set up the ISAKMP Security Associations, Host-A's next step is to initiate the Oakley phase 2 message exchanges (also known as Oakley Quick Mode) to define the Security Associations and keys that will be used to protect IP datagrams exchanged between the pair of users. (In the Internet drafts, these are referred to somewhat obtusely as “non-ISAKMP SAs.”)

Because the purpose of the phase 1 negotiations was to agree on how to protect ISAKMP messages, all ISAKMP phase 2 payloads, but not the ISAKMP header itself, must be encrypted using the algorithm agreed to by the phase 1 negotiations.

When Oakley Quick Mode is used in phase 2, authentication is achieved through the use of several cryptographically based hash functions. The input to the hash functions comes partly from phase 1 information (SKEYID) and partly from information exchanged in phase 2. Phase 2 authentication is based on certificates, but the phase 2 process itself does not use certificates directly. Instead, it uses the SKEYID_a material from phase 1, which itself was authenticated through certificates.

Oakley Quick Mode comes in two forms:

- Without a Key Exchange attribute, Quick Mode can be used to refresh the cryptographic keys, but does not provide the property of Perfect Forward Secrecy (PFS).
- With a Key Exchange attribute, Quick Mode can be used to refresh the cryptographic keys in a way that provides PFS. This is accomplished by including an exchange of public Diffie-Hellman values within messages 1 and 2.

Note: PFS apparently is a property that is very much desired by cryptography experts, but strangely enough, the specifications treat PFS as optional. They mandate that a system must be capable of handling the Key Exchange field when it is present in a Quick Mode message, but do not require a system to include the field within the message.

A detailed description of the phase 2 messages and exchanged information follows.

IKE phase 2, message 1

Message 1 of a Quick Mode Exchange allows Host-A to authenticate itself, to select a nonce, to propose Security Associations to Host-B, to execute an exchange of public Diffie-Hellman values, and to indicate if it is acting on its own behalf or as a proxy negotiator for another entity. An overview of the format of message 1 is shown in Figure 22-41.

Note: Inclusion of a Key Exchange field is optional. However, when Perfect Forward Secrecy is used, it must be present.

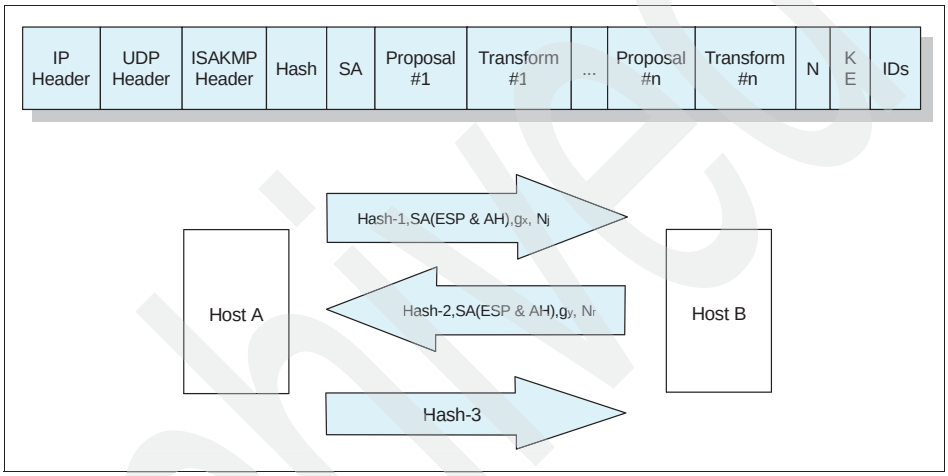


Figure 22-41 Message 1 of an ISAKMP phase 2 Quick Mode Exchange

Because we assumed that Host-A and Host-B are each acting on their own behalf, the user identity fields illustrated in Figure 22-41 will not be present. The message will consist of:

ISAKMP header The ISAKMP header indicates an exchange type of Quick Mode, includes a non-zero Message ID chosen by Host-A, includes the initiator and responder cookie values chosen in phase 1 (that is, Cookie-A and Cookie-B), and turns on the encryption flag to indicate that the payloads of the ISAKMP message are encrypted according to the algorithm and key negotiated during phase 1.

Hash

A Hash payload must immediately follow the ISAKMP header. HASH_1 uses the keyed pseudo-random function that was negotiated during the phase 1 exchanges, and is derived from the following information:

$$HASH\_1 = prf(SKEYID\_a, M\text{-}ID, SA, N_{qmi}, KE, ID_{qmi}, ID_{qmr})$$

- SKEYID_a was derived from the phase 1 exchanges.
- M-ID is the message ID of this message.
- SA is the Security Association payload carried in this message, including all proposals that were offered.
- Nonce is a new value different from the one used in phase 1.
- KE is the public Diffie-Hellman value carried in this message. This quantity is chosen by Host-A, and is denoted as g_{qm}^{x}. Note that this is not the same quantity as g^{x} that was used in the phase 1 exchanges.
- IDs, which can identify either the endpoints of the phase 1 exchange or endpoints on whose behalf the protocol SA should be negotiated (proxy IDs when IKE is used in client mode). These can subsequently be different from the IDs used in phase 1.

Note: The use of KE and ID is optional, depending if PFS is used.

Security Association

Indicates IP as the Domain of Interpretation.

Proposal, Transform Pairs

There can be one or more of these pairs in this message. The first proposal payload is numbered 1, identifies an IPSec protocol to be used, and includes an SPI value that is randomly chosen by Host-A for use with that protocol. The proposal payload is followed by a single transform payload that indicates the cryptographic algorithm to be used with that protocol. The second proposal payload is numbered 2, and so on.

Nonce payload

This contains the nonce N^{qmi} that was chosen randomly by Host-A.

KE This is the key exchange payload that carries the public Diffie-Hellman value chosen by Host-A, g_{qm}^{x}. There is also a field called Group that indicates the prime number and generator used in the Diffie-Hellman exchange.

ID payload Specifies the endpoints for this SA.

IKE phase 2, message 2

After Host-B receives message 1 from Host-A and successfully authenticates it using HASH_1, it constructs a reply, message 2, to be sent back to Host-A. The Message ID of the reply is the same one that Host-A used in message 1.

Host-B chooses new values for the following:

Hash The hash payload now carries the value HASH_2, which is defined as:

$$HASH\_2 = prf(SKEYID\_a, N_{qmi}, M\text{-}ID, SA, N_{qmr}, KE, ID_{qmi}, ID_{qmr})$$

Security Association The Security Association payload only describes the single chosen proposal and its associated transforms, not all of the protection suites offered by Host-A. Host-B also chooses an SPI value for the selected protocol. Host-B's SPI does not depend in any way on the SPI that Host-A assigned to that protocol when it offered the proposal. That is, it is not necessary that SPI_A be the same as SPI_B; it is only necessary that they each be non-zero and that they each be randomly chosen.

Nonce Nonce payload now carries N_r, a random value chosen by Host-B.

KE Key exchange payload now carries Host-B's public Diffie-Hellman value, g_{qm}^{y}.

At this point, Host-A and Host-B have exchanged nonces and public Diffie-Hellman values. Each one can use this in conjunction with other information to derive a pair of keys, one for each direction of transmission.

Generating the keys (phase 2)

Using the nonces, public Diffie-Hellman values, SPIs, protocol code points exchanged in messages 1 and 2 of phase 2, and the SKEYID value from phase 1, each host now has enough information to derive two sets of keying material:

- When PFS is used:
  - For data generated by Host-A and received by Host-B, the keying material is:

    $KEYMAT_{AB} = prf(SKEYID\_d, g_{qm}^{xy}, protocol, SPI_B, N_{qmi}, N_{qmr})$
  - For data generated by Host-B and received by Host-A, the keying material is:

    $KEYMAT_{BA} = prf(SKEYID\_d, g_{qm}^{xy}, protocol, SPI_A, N_{qmi}, N_{qmr})$
- When PFS is not used:
  - For data generated by Host-A and received by Host-B, the keying material is:

    $KEYMAT_{AB} = prf(SKEYID\_d, protocol, SPI_B, N_{qmi}, N_{qmr})$
  - For data generated by Host-B and received by Host-A, the keying material is:

    $KEYMAT_{BA} = prf(SKEYID\_d, protocol, SPI_A, N_{qmi}, N_{qmr})$

Note: Depending on the particular case, Host-A might need to derive multiple keys for the following purposes:

- Generating the integrity check value for transmitted datagrams
- Validating the integrity check value of received datagrams
- Encrypting transmitted datagrams
- Decrypting received datagrams

Likewise, Host-B needs to derive the mirror image of the same keys. For example, the key that Host-B uses to encrypt its outbound messages is the same key that Host-A uses to decrypt its inbound messages, and so on.

IKE phase 2, message 3

At this point, Host-A and Host-B have exchanged all the information necessary for them to derive the necessary keying material. The third message in the Quick Mode exchange is used by Host-A to prove its alive state, which it does by producing a hash function that covers the message ID and both nonces that were exchanged in messages 1 and 2. Message 3 consists only of the ISAKMP header and a hash payload that carries:

$HASH\_3 = prf(SKEYID\_a, 0, M\text{-}ID, N_{qmi}, N_{qmr})$

When Host-B receives this message and verifies the hash, both systems can begin to use the negotiated security protocols to protect their user data streams.

Negotiating multiple Security Associations

It is also possible to negotiate multiple Security Associations, each with its own set of keying material, within a single three-message Quick Mode exchange.

The message formats are very similar to the previously illustrated ones, so we only highlight the differences:

- Message 1 carries multiple Security Association payloads, each offering a range of protection suites.
- HASH_1 covers the entire set of all offered Security Associations carried in message 1. That is, each Security Association and all of its offered proposals are included.
- In message 2, for each offered SA, Host-B selects a single protection suite. That is, if n SAs are open for negotiation, Host-B chooses n protection suites, one from each proposal.
- As was the case for HASH_1, HASH_2 now covers the entire set of all offered Security Associations carried in message 1. That is, each Security Association and all of its offered proposals are included.
- After messages 1 and 2 have been exchanged, Host-A and Host-B generate the keying material for each of the accepted protection suites, using the same formulas as in “Generating the keys (phase 2)” on page 844, applied individually for each accepted SA. Even though the nonces and the public Diffie-Hellman values are the same for all selected suites, the keying material derived for each selected protection suite is different because each proposal has a different SPI.
- Because multiple Security Associations have been negotiated, it is a matter of local choice as to which one is used to protect a given datagram. A receiving system must be capable of processing a datagram that is protected by any SA that has been negotiated. That is, it is legal for a given source host to send two consecutive datagrams to a destination system, where each datagram was protected by a different SA.

Using IKE with remote access

The critical element in the remote access scenario is the use of Oakley to identify the remote host by name, rather than by its dynamically assigned IP address. After the remote host's identity has been authenticated and the mapping to its dynamically assigned IP address has been ascertained, the remainder of the processes are the same as we have described for the other scenarios. For example, if the corporate intranet is considered to be trusted, the remote host needs to establish a single SA between itself and the firewall. But if the corporate

intranet is considered to be untrusted, it might be necessary for the remote host to set up two SAs: one between itself and the firewall, and a second between itself and the destination host.

Recall that a single ISAKMP phase 1 negotiation can protect several subsequent phase 2 negotiations. Phase 1 ISAKMP negotiations use computationally intensive public key cryptographic operations, while phase 2 negotiations use the less computationally intensive symmetric key cryptographic operations. Therefore, the heavy computational load only occurs in phase 1, which is only executed when the dial-up connection is first initiated.

The principal points that pertain to the remote access case are:

- The remote host's dynamically assigned address is the one that is placed in the IP header of all ISAKMP messages.
- The remote host's permanent identifier (such as an e-mail address) is the quantity that is placed in the ID field of the ISAKMP phase 1 messages.
- The remote host's certificate used in the ISAKMP exchange must be associated with the remote host's permanent identifier.
- In traffic-bearing datagrams, the remote host's dynamically assigned IP address is used. This is necessary because the destination IP address that appears in the datagram's IP header is used in conjunction with the SPI and protocol type to identify the relevant IPSec Security Association for processing the inbound datagram.

22.5 SOCKS

SOCKS is a standard for circuit-level gateways. It does not require the processing costs associated with a more conventional proxy server where a user has to consciously connect to the firewall first before requesting the second connection to the destination (see Figure 22-42 on page 847).

The user starts a client application with the destination server IP address. Instead of directly starting a session with the destination server, the client initiates a session to the SOCKS server on the firewall. The SOCKS server then validates that the source address and user ID are permitted to establish an onward connection into the nonsecure network, and then creates the second session.

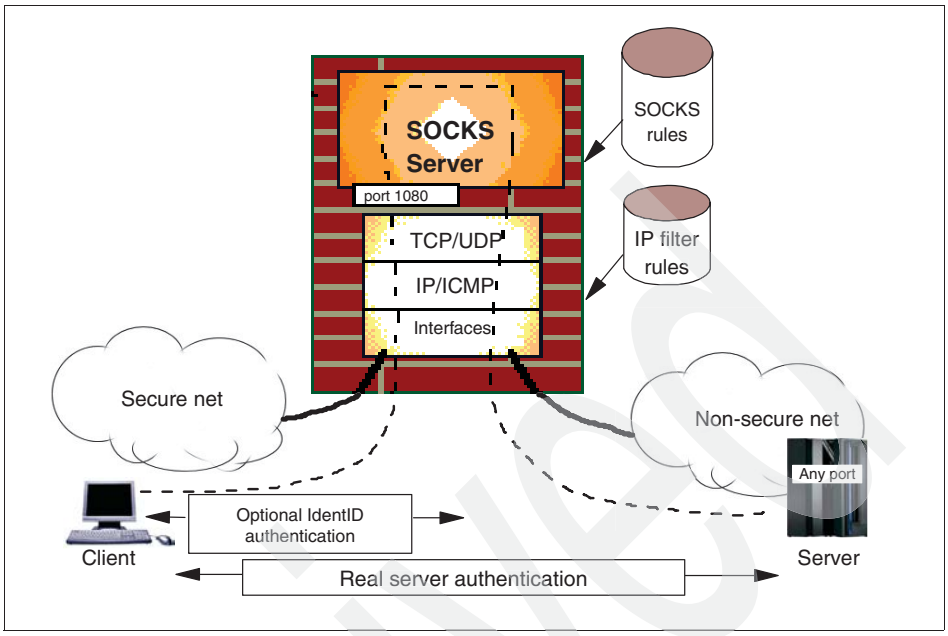


Figure 22-42 SOCKS server

SOCKS needs to have new versions of the client code (called SOCKS-enabled clients) and a separate set of configuration profiles on the firewall. However, the server machine does not need modification; indeed, it is unaware that the session is being relayed by the SOCKS server. Both the client and the SOCKS server need to have SOCKS code. The SOCKS server acts as an application-level router between the client and the real application server. SOCKSv4 is for outbound TCP sessions only. It is simpler for the private network user, but does not have secure password delivery, so it is not intended for sessions between public network users and private network applications. SOCKSv5 provides for several authentication methods and can therefore be used for inbound connections as well, though these need to be used with caution. SOCKSv5 also supports UDP-based applications and protocols.

The majority of Web browsers are SOCKS-enabled and you can get SOCKS-enabled TCP/IP stacks for most platforms. For additional information, refer to RFC 1928, RFC 1929, RFC 1961, and the following URL:

`http://www.socks.nec.com`

22.5.1 SOCKS Version 5 (SOCKSv5)

SOCKS version 5 is a proposed standard protocol with a status of elective. It is described in RFC 1928.

Application-level gateways provide secure connections for some applications such as Telnet, FTP, and SMTP. However, it is not easy to write proxy code for each new application. Generally, the proxy service becomes available after some time, even if the service can be used directly and application-level gateways do not allow UDP connections. SOCKSv5 satisfies all these shortcomings and requirements with a strong authentication mechanism and the hiding of addresses from a non-secure network. Although supporting UDP might seem to be vulnerable, it can be configured to pass UDP for particular users and particular applications only.

The SOCKSv5 concept is based on SOCKSv4 with some extensions such as UDP support, new and various sophisticated authentication methods, and extended addressing schemes to cover domain-name and IPv6. SOCKSv5 supports a range of authentication methods, including:

- User name/password authentication
- One-time password generators
- Kerberos
- Remote Authentication Dial-In User Services (RADIUS)
- Password Authentication Protocol (PAP)
- IPSec authentication method

SOCKSv5 also supports the following encryption standards:

- DES
- Triple DES
- IPSec

The following tunneling protocols are supported:

- PPTP
- L2F
- L2TP

The following key management systems are supported:

- SKIP
- ISAKMP/Oakley

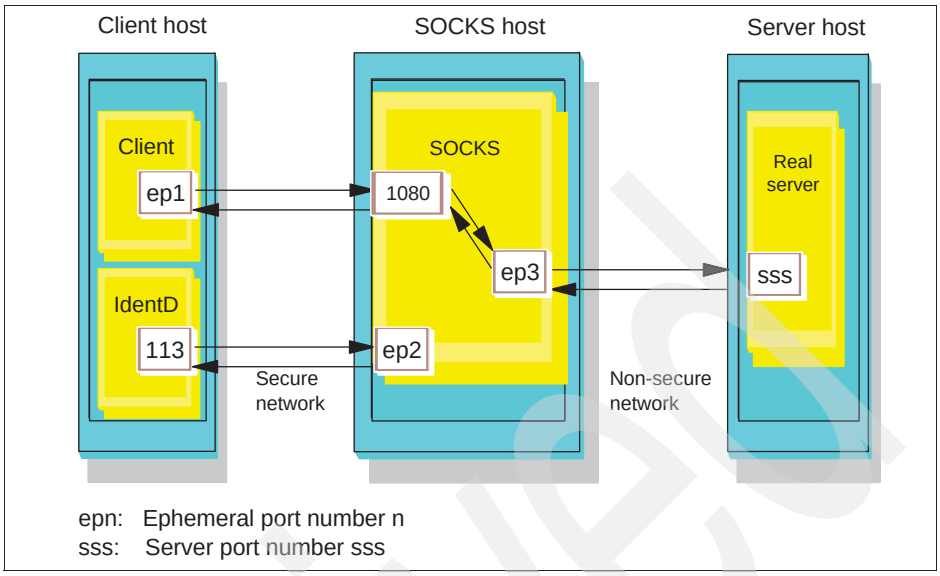


Figure 22-43 Socks TCP segment flow

The SOCKSv5 server listens for connections on a given port, usually 1080. According to the connection type (TCP or UDP), the steps discussed in the following sections establish a connection.

SOCKSv5 TCP connection

To establish a connection using TCP, the client first sends a TCP packet that contains session request information through port 1080 to the server (see Figure 22-43). If the access permissions allow this operation and the connection request succeeds, the client enters an authentication negotiation. In this state, the authentication type is determined, after which the client sends a relay request. The SOCKSv5 server evaluates the request and either establishes the connection or rejects it. The client sends the following message, which contains a version identifier and method options (Figure 22-44).

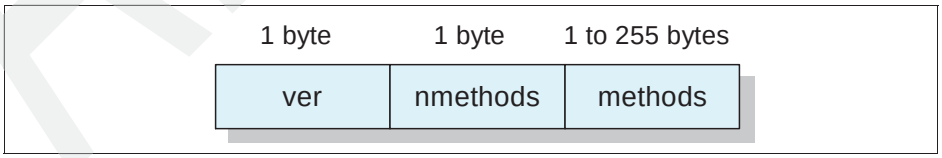


Figure 22-44 SOCKSv5: Version identifier and method selection message format

Where:

VER Indicates the version of SOCKS. For SOCKSv5, the value is hexadecimal X'05'.

NMETHODS Indicates the number of the methods in the methods field.

METHODS Indicates the supported authentication and encapsulation methods.

The server responds by the following message (Figure 22-45).

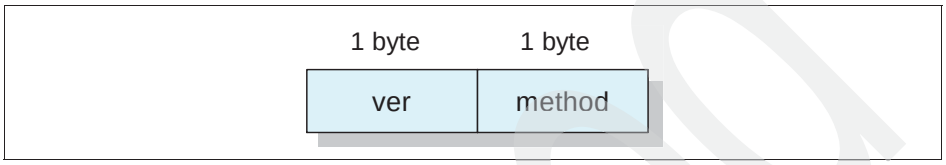


Figure 22-45 SOCKSv5: Selected method message format

The hexadecimal values for current standard methods are as follows:

X'00' No authentication required

X'01' GSSAPI

X'02' User name/password

X'03' to X'7F' IANA assigned

X'80' to X'FE' Reserved for private methods

X'FF' No acceptable methods

All implementations should support user name/password and GSSAPI authentication methods.

SOCKSv5 Connect

After authentication completes successfully, the client sends the request details. If an encapsulation method is negotiated during the method negotiation, the selected encapsulation method must be applied for the following messages. The detail request message format issued by the client is as shown in Figure 22-46.

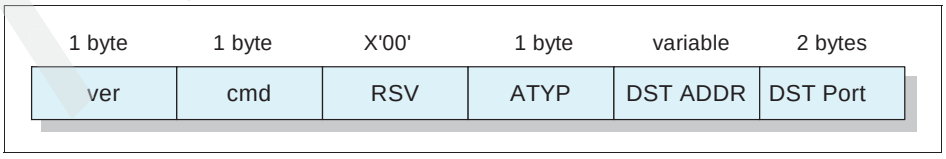


Figure 22-46 SOCKSv5: Detail request message format

Where:

VER Socks protocol version. For SOCKSv5, the value is hexadecimal X'05'.

CMD SOCKS command in octets:

- **X'01'** Connect
- **X'02'** BIND
- **X'03'** UDP associate

RSV Reserved for future use.

ATYP Address types in octets:

- **X'01'** IPv4 address
- **X'03'** Domain-name
- **X'04'** IPv6 address

DST.ADDR Desired destination address.

DST.PORT Desired destination port in network octet order.

An IPv4 address is stored as 4 bytes. An IPv6 address is stored as 16 bytes.

A domain name is stored as a length byte, and then a fully qualified domain name. There is no trailing null at the end of the domain name.

The server evaluates the request detail message and replies with one or more messages. Here is the reply message format issued by the server (Figure 22-47).

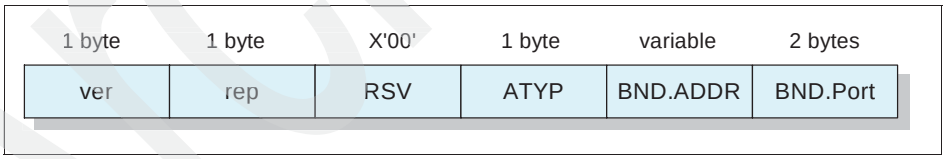


Figure 22-47 SOCKSv5: Server reply message format

Where:

VER Socks protocol version. For SOCKSv5, the value is hexadecimal X'05'.

REP Reply field:

- **X'00'** Succeeded
- **X'01'** General SOCKS server failure
- **X'02'** Connection not allowed by ruleset

X'03'	Network unreachable
X'04'	Host unreachable
X'05'	Connection refused
X'06'	TTL expired
X'07'	Command not supported
X'08'	Address type not supported
X'09' to X'FF'	Unassigned
RSV	Reserved for future use.
ATYP	Address types in octets:
X'01'	IPv4 address
X'03'	Domain name
X'04'	IPv6 address
BND.ADDR	Server bound address.
BND.PORT	Server bound port in network octet order.

SOCKSv5 BIND

To accept an incoming connection from the Internet, use the same request and reply format as described earlier for SOCKSv5 Connect, setting the CMD field to BIND. However, you receive two reply packets.

The first reply contains the IP address and port number on which the SOCKS server has put a listener.

When the remote system calls into the SOCKS server, you get a second reply with the BND.ADDR and BIND.Port fields containing details of the remote server.

SOCKSv5 UDP connection

To be able use a UDP connection over a SOCKS server, the client first issues the UDP ASSOCIATE command to the SOCKSv5 server. The SOCKSv5 server then assigns a UDP port to which the client sends all UDP datagrams. Each UDP datagram has a UDP request header. The UDP request header format is as follows (Figure 22-48).

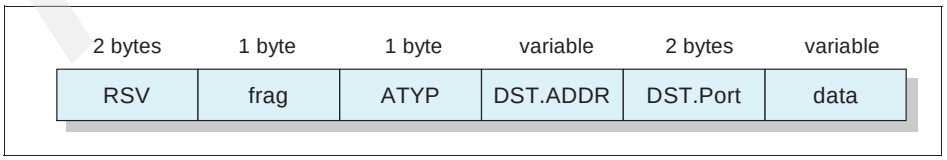


Figure 22-48 SOCKSv5: UDP datagram request header format

Where:

RSV	Reserved for future use. All bytes are zero.
FRAG	Current fragment number.
ATYP	Address types in octets:
X'01'	IPv4 address
X'03'	Domain-name
X'04'	IPv6 address
DST.ADDR	Desired destination address.
DST.PORT	Desired destination port in network octet order.
DATA	User data.

The UDP relay server gets the IP address of the client, which sends UDP datagrams to the port specified by DST.PORT. It then discards any datagram that comes from another source.

22.6 Secure Shell (1 and 2)

SSH can secure connections between systems. It allows application traffic, such as that generated by Telnet, FTP POP3, or even X Window System, to be both encrypted and compressed. Compression is useful over slow modem links. Implementations allow the user a choice of encryption methods.

Client software often offers both SSH1 and SSH2 support. The user is authenticated by password or public/private key.

SSH1 offers Blowfish, DES, 3DES, and RC4 encryption ciphers.

SSH2 offers 3DES, RC4, and Twofish encryption ciphers.

22.6.1 SSH overview

SSH establishes a single TCP/IP connection from the client to the server. The traffic sent down this connection is encrypted, and optionally compressed using LempleZiv compression. Public/private keys can be used to verify both the user and the identity of the remote system.

SSH and X Window System

X Window System sessions can pass through the SSH connection. The SSH server generates a new DISPLAY variable (and xauth key) for the remote

Archived

X Window System's clients. SSH forwards the X Window System traffic to the user's local X Server. Users have to supply their own X Server applications; make sure it is listening on the local host.

SSH port forwarding

SSH offers the ability to map TCP/IP ports across systems. For example, you can configure SSH to copy data between a port on the client's local host and the servers POP3 port. By running a POP3 client and pointing it at the local host, you establish a secure encrypted session over which to read e-mail.

22.7 Secure Sockets Layer (SSL)

SSL is a security protocol that was developed by Netscape Communications Corporation, along with RSA Data Security, Inc. The primary goal of the SSL protocol is to provide a private channel between communicating applications, which ensures privacy of data, authentication of the partners, and integrity.

22.7.1 SSL overview

SSL provides an alternative to the standard TCP/IP socket API that has security implemented within it. Therefore, in theory, it is possible to run any TCP/IP application in a secure way without changing the application. In practice, SSL is only widely implemented for HTTP connections, but Netscape Communications Corp. has stated an intention to employ it for other application types, such as NNTP and Telnet, and there are several such implementations freely available on the Internet. IBM, for example, uses SSL to enhance security for TN3270 sessions in the IBM WebSphere Host On-Demand and eNetwork Communications Server products.

SSL is composed of two layers:

- At the lower layer, a protocol for transferring data using a variety of predefined cipher and authentication combinations, called the *SSL Record Protocol*. Figure 22-49 on page 855 illustrates this and contrasts it with a standard HTTP socket connection. Note that this diagram shows SSL as providing a simple socket interface on which other applications can be layered. In reality, current implementations have the socket interface embedded within the application and do not expose an API that other applications can use.
- On the upper layer, a protocol for initial authentication and transfer of encryption keys, called the *SSL Handshake Protocol*.

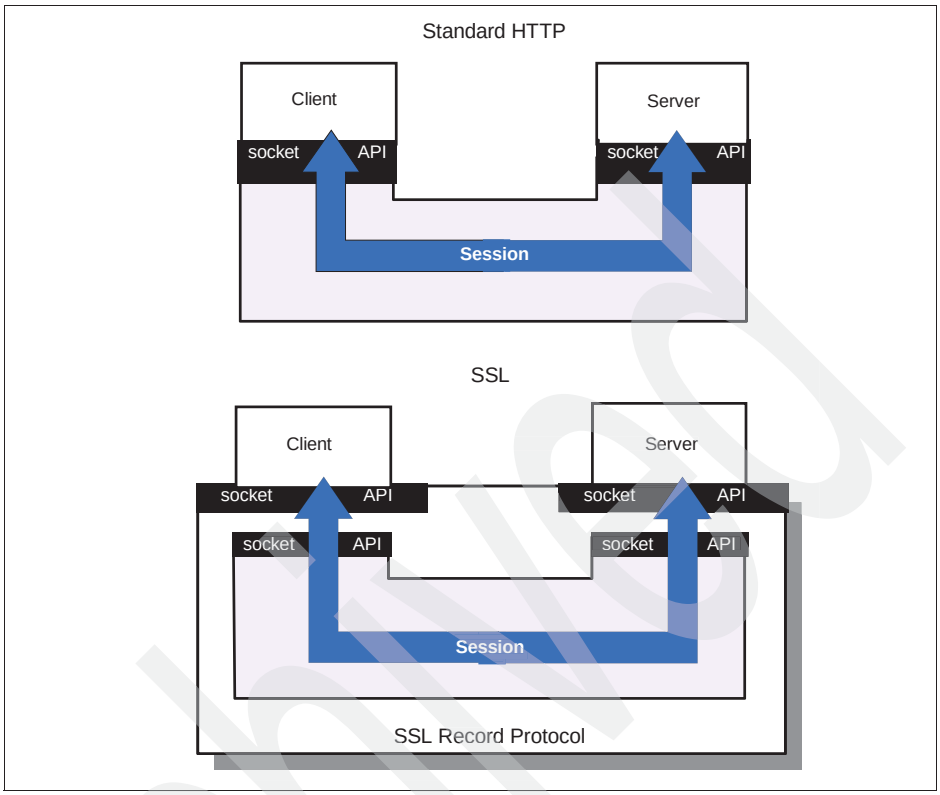


Figure 22-49 SSL: Comparison of standard and SSL sessions

An SSL session is initiated as follows:

1. On the client (browser), the user requests a document with a special URL that starts with https: instead of http:, either by typing it into the URL input field, or by clicking a link.
2. The client code recognizes the SSL request and establishes a connection through TCP port 443 to the SSL code on the server.
3. The client then initiates the SSL handshake phase, using the SSL Record Protocol as a carrier. At this point, there is no encryption or integrity checking built in to the connection.

The SSL protocol addresses the following security issues:

Privacy After the symmetric key is established in the initial handshake, the messages are encrypted using this key.

Integrity Messages contain a message authentication code (MAC) ensuring the message integrity.

Authentication During the handshake, the client authenticates the server using an asymmetric or public key. It can also be based on certificates.

SSL requires that each message is encrypted and decrypted and therefore has a high performance and resource cost.

Differences between SSL V2.0 and SSL V3.0

There is backward compatibility between SSL V2.0 and SSL V3.0. An SSL V3.0 server implementation should be able accept the connection request from an SSL V2.0 client. The main differences between SSL V2.0 and SSL V3.0 are as follows:

- SSL V2.0 does not support client authentication.
- SSL V3.0 supports more ciphering types in the CipherSpec.

22.7.2 SSL protocol

The SSL protocol is located at the top of the transport layer. SSL is also a layered protocol itself. It simply takes the data from the application layer, reformats it, and transmits it to the transport layer. SSL handles a message as follows:

- The sender performs the following tasks:
  a. Takes the message from upper layer.
  b. Fragments the data to manageable blocks.
  c. Optionally compresses the data.
  d. Applies a message authentication code (MAC).
  e. Encrypts the data.
  f. Transmits the result to the lower layer.
- The receiver performs the following tasks:
  a. Takes the data from lower layer.
  b. Decrypts.
  c. Verifies the data with the negotiated MAC key.
  d. Decompresses the data if compression was used.
  e. Reassembles the message.
  f. Transmits the message to the upper layer.

An SSL session works in different states. These states are *session* and *connection* states. The SSL handshake protocol (see “SSL handshake protocol” on page 858) coordinates the states of the client and the server. In addition, there are read and write states defined to coordinate the encryption according to the change CipherSpec messages.

When either party sends a change CipherSpec message, it changes the pending write state to current write state. Again, when either party receives a change CipherSpec message, it changes the pending read state to the current read state.

The session state includes the following components:

Session identifier	An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
Peer certificate	Certificate of the peer. This field is optional; it can be empty.
Compression method	The compression algorithm.
CipherSpec	Specifies data encryption algorithm (such as null, DES) and a MAC algorithm.
Master secret	48-byte shared secret between the client and the server.
Is resumable	A flag indicating whether the session can be used for new connections.

The connection state includes the following components:

Server and client random	An arbitrary byte sequence chosen by the client and server for each connection.
Server write MAC secret	The secret used for MAC operations by the server.
Client write MAC secret	The secret used for MAC operations by the client.
Server write key	The cipher key for the server to encrypt the data and the client to decrypt the data.
Client write key	The cipher key for the client to encrypt the data and the server to decrypt the data.
Initialization vectors	Initialization vectors store the encryption information.
Sequence numbers	A sequence number indicates the number of the message transmitted since the last change CipherSpec message. Both the client and the server maintain sequence numbers.

Change CipherSpec protocol

The change CipherSpec protocol is responsible for sending change CipherSpec messages. At any time, the client can request to change current cryptographic parameters such as the handshake key exchange. Following the change CipherSpec notification, the client sends a handshake key exchange and if available, certificate verify messages, and the server sends a change

Archived

CipherSpec message after processing the key exchange message. After that, the newly agreed keys will be used until the next change CipherSpec request. The change CipherSpec message is sent after the hello messages during the negotiation.

SSL handshake protocol

The SSL handshake protocol allows the client and server to determine the required parameters for an SSL connection such as protocol version, cryptographic algorithms, optional client or server authentication, and public key encryption methods to generate shared secrets. During this process, all handshake messages are forwarded to the SSL record layer to be encapsulated into special SSL messages. Figure 22-50 illustrates an SSL handshake process.

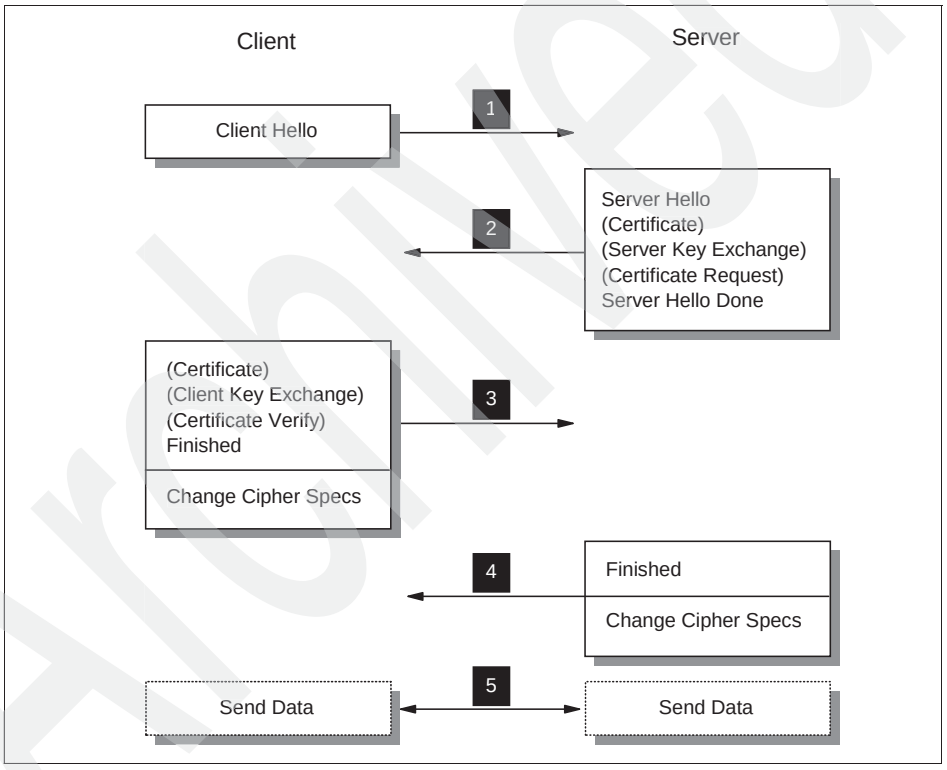


Figure 22-50 SSL: Handshake process

We explain the SSL handshake process detailed in Figure 22-50 in more detail:

1. The client sends a connection request with a client hello message. This message includes:
   – Desired version number.

- Time information (the current time and date in standard UNIX 32-bit format).
- Optionally, a session ID. If it is not specified the server will try to resume previous sessions or return an error message.
- Cipher suites. (List of the cryptographic options supported by the client. These are authentication modes, key exchange methods, encryptions, and MAC algorithms.)
- Compression methods supported by the client.
- A random value.

2. The server evaluates the parameters sent by the client hello message and returns a server hello message that includes the following parameters that were selected by the server to be used for the SSL session:
   - Version number
   - Time information (the current time and date in standard UNIX 32-bit format)
   - Session ID
   - Cipher suite
   - Compression method
   - A random value

   Following the server hello message, the server sends the following messages:
   - Server certificate if the server is required to be authenticated
   - A server key exchange message if there is no certificate available or the certificate is for signing only
   - A certificate request if the client is required to be authenticated

   Finally, the server sends a server hello done message and begins to wait for the client response.
3. The client sends the following messages:
   - If the server has sent a certificate request, the client must send a certificate or a no certificate message.
   - If the server has sent a server key exchange message, the client sends a client key exchange message based on the public key algorithm determined with the hello messages.
   - If the client has sent a certificate, the client verifies the server certificate and sends a certificate verify message indicating the result.

Archived

The client then sends a finished message indicating the negotiation part is completed. The client also sends a change CipherSpec message to generate shared secrets. Note that this is not controlled by the handshake protocol; the change CipherSpec protocol manages this part of the operation.

4. The server sends a finished message indicating that the negotiation part is completed. The server then sends the change CipherSpec message.
5. Finally, the session partners separately generate an encryption key, in which they derive the keys to use in the encrypted session that follows from the master key. The handshake protocol changes the state to the connection state. All data taken from the application layer is transmitted as special messages to the other party.

There are significant additional processing costs associated with starting up an SSL session compared with a normal HTTP connection. The protocol avoids some of these costs by allowing the client and server to retain session key information and to resume that session without negotiating and authenticating a second time.

Following the handshake, both session partners have generated a master key. From that key, they generate other session keys, which are used in the symmetric-key encryption of the session data and in the creation of message digests. The first message encrypted in this way is the finished message from the server. If the client can interpret the finished message, it means:

- Privacy has been achieved, because the message is encrypted using a symmetric-key bulk cipher (such as DES or RC4).
- The message integrity is assured, because it contains a message authentication code (MAC), which is a message digest of the message itself plus material derived from the master key.
- The server has been authenticated, because it was able to derive the master key from the pre-master key. Because this was sent using the server's public key, it can only be decrypted by the server (using its private key). Note that this relies on the integrity of the server's public key certificate.

SSL record protocol

After the master key has been determined, the client and server can use it to encrypt application data. The SSL record protocol specifies a format for these messages. In general, they include a message digest to ensure that they have not been altered and the whole message is encrypted using a symmetric cipher. Usually, this uses the RC2 or RC4 algorithm, although DES, triple-DES, and IDEA are also supported by the specification.

The U.S. National Security Agency (NSA), a department of the United States federal government, imposed restrictions on the size of the encryption key that

can be used in software exported outside the U.S. These rules have been reviewed, but originally the key was limited to an effective size of 56 bits. The RC2 and RC4 algorithms achieved this by using a key in which all but 56 bits are set to a fixed value. International (export) versions of software products had this hobbled security built into them. SSL checks for mismatches between the export and nonexport versions in the negotiation phase of the handshake. For example, if a U.S. browser tries to connect with SSL to an export server, they will agree on export-strength encryption. See 22.2.7, “Export/import restrictions on cryptography” on page 793 for more information about recent changes of U.S. export regulations of cryptographic material.

22.8 Transport Layer Security (TLS)

The Transport Layer Security 1.0 protocol is based on SSL. The TLS 1.0 protocol is documented in RFC 2246. Two applications (without knowing each other’s code) can use TLS to communicate securely. There are no significant differences between SSL 3.0 and TLS 1.0. They can interoperate with some modifications of the message formats. A TLS 1.0 application can back down to an SSL 3.0 connection.

22.9 Secure Multipurpose Internet Mail Extension (S-MIME)

Secure Multipurpose Internet Mail Extension (S-MIME) can be thought of as a very specific SSL-like protocol. S-MIME is an application-level security construct, but its use is limited to protecting e-mail through encryption and digital signatures. It relies on public key technology, and uses X.509 certificates to establish the identities of the communicating parties. S-MIME can be implemented in the communicating end systems; it is not used by intermediate routers or firewalls.

22.10 Virtual private networks (VPNs) overview

The Internet has become a popular, low-cost backbone infrastructure. Its universal reach has led many companies to consider constructing a secure virtual private network (VPN) over the public Internet. The challenge in designing a VPN for today's global business environment will be to exploit the public Internet backbone for both intra-company and inter-company communication while still providing the security of the traditional private, self-administered corporate network.

In this chapter, we begin by defining a virtual private network (VPN) and explaining the benefits that clients can achieve from its implementation. After discussing the security considerations and planning aspects, we then describe the VPN solutions available in the market today.

22.10.1 VPN introduction and benefits

With the explosive growth of the Internet, companies are beginning to ask: “How can we best exploit the Internet for our business?” Initially, companies were using the Internet to promote their company's image, products, and services by providing World Wide Web access to corporate Web sites. Today, however, the Internet potential is limitless, and the focus has shifted to e-business, using the global reach of the Internet for easy access to key business applications and data that reside in traditional IT systems. Companies can now securely, and cost-effectively, extend the reach of their applications and data across the world through the implementation of secure virtual private network (VPN) solutions.

A virtual private network (VPN) is an extension of an enterprise's private intranet across a public network such as the Internet, creating a secure private connection, essentially through a private *tunnel*. VPNs securely convey information across the Internet connecting remote users, branch offices, and business partners into an extended corporate network, as shown in Figure 22-51 on page 863. Internet service providers (ISPs) offer cost-effective access to the Internet (via direct lines or local telephone numbers), enabling companies to eliminate their current, expensive leased lines, long-distance calls, and toll-free telephone numbers.

Archive

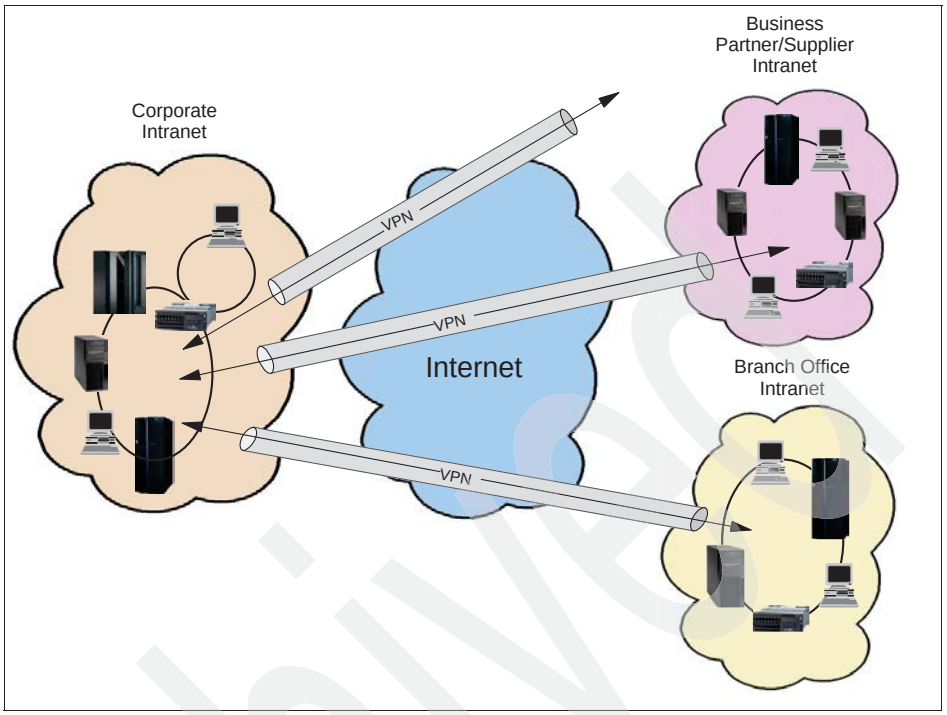


Figure 22-51 Virtual private networks

A 1997 VPN Research Report, by Infonetics Research, Inc., estimates savings from 20% to 47% of wide area network (WAN) costs by replacing leased lines to remote sites with VPNs. And, for remote access VPNs, savings can be 60% to 80% of corporate remote access dial-up costs. Additionally, Internet access is available worldwide where other connectivity alternatives might not be available.

The technology to implement these virtual private networks, however, is just becoming standardized. Some networking vendors today are offering nonstandards-based VPN solutions that make it difficult for a company to incorporate all its employees and business partners/suppliers into an extended corporate network. However, VPN solutions based on Internet Engineering Task Force (IETF) standards will provide support for the full range of VPN scenarios with more interoperability and expansion capabilities.

The key to maximizing the value of a VPN is the ability for companies to evolve their VPNs as their business needs change and to easily upgrade to future TCP/IP technology. Vendors that support a broad range of hardware and software VPN products provide the flexibility to meet these requirements. VPN solutions today run mainly in the IPv4 environment, but it is important that they have the capability of being upgraded to IPv6 to remain interoperable with your

business partner's and supplier's VPN solutions. Perhaps equally critical is the ability to work with a vendor that understands the issues of deploying a VPN. The implementation of a successful VPN involves more than technology. The vendor's networking experience plays heavily into this equation.

22.11 Kerberos authentication and authorization system

The Kerberos Network Authentication Service Version 5 is a *proposed standard protocol*. Its status is *elective* and described in RFC 1510.

According to *The Enlarged Devil's Dictionary*, by Ambrose Bierce, Kerberos is "the watchdog of Hades, whose duty it was to guard the entrance against whom or what does not clearly appear; Kerberos is known to have had three heads."

A Kerberos service is normally run on its own system in a secure area. Users have to validate themselves to Kerberos before they are allowed to connect to other servers in the network. The server's identities can also be checked against Kerberos.

The Kerberos Authentication and Authorization System is an encryption-based security system that provides mutual authentication between the users and the servers in a network environment. The assumed goals for this system are:

- Authentication to prevent fraudulent requests/responses between users and servers that must be confidential and between groups of at least one user and one service.
- Authorization can be implemented independently from the authentication by each service that wants to provide its own authorization system. The authorization system can assume that the authentication of a user/client is reliable.
- Permits the implementation of an accounting system that is integrated, secure, and reliable, with modular attachment and support for "chargebacks" or billing purposes.

The Kerberos system is mainly used for authentication purposes, but it also provides the flexibility to add authorization information.

Archive

22.11.1 Assumptions

Kerberos assumes the following:

- The environment using this security system includes public and private workstations that can be located in areas with minimal physical security, a campus network without link encryption that can be composed of dispersed local networks connected by backbones or gateways, centrally operated servers in locked rooms with moderate physical security, and centrally operated servers with considerable physical security.
- Confidential data or high-risk operations such as a bank transaction cannot be part of this environment without additional security, because after you have a workstation as a terminal, you can emulate certain conditions and normal data will be flowing without any encryption protection.
- One of the cryptosystems used is the Data Encryption Standard (DES), which has been developed to be modular and replaceable by the Kerberos designers.
- Kerberos assumes a loosely synchronized clock in the whole system, so the workstation has to have a synchronization tool such as the time server provided.

22.11.2 Naming

A *principal identifier* is the name that identifies a client or a service for the Kerberos system.

In Version 4, the identifier consists of three components:

- The *principal* name is unique for each client and service assigned by the Kerberos Manager.
- The *instance* name used for distinct authentication is an added label for clients and services, which exist in several forms. For users, an instance can provide different identifiers for different privileges. For services, an instance usually specifies the host name of the machine that provides this service.
- The *realm* name used to allow independently administered Kerberos sites. The principal name and the instance are qualified by the realm to which they belong, and are unique only within that realm. The realm is commonly the domain name.

In Version 4, each of the three components has a limit of 39 characters long. Due to conventions, the period (.) is not an acceptable character.

In Version 5, the identifier consists of two parts only, the *realm* and the *remainder*, which is a sequence of however many components are needed to

name the principal. Both the realm and each component of the remainder are defined as ASN.1 (Abstract Syntax Notation One, ISO standard 8824) GeneralStrings. This puts few restrictions on the characters available for principal identifiers.

22.11.3 Kerberos authentication process

In the Kerberos system, a client that wants to contact a server for its service, first has to ask for a *ticket* from a mutually trusted third party, the Kerberos Authentication Server (KAS). This ticket is obtained as a function where one of the components is a private key known only by the service and the Kerberos Authentication Server so that the service can be confident that the information on the ticket originates from Kerberos. The client is known to the KAS as a principal name (c). The private key (K_c) is the authentication key known only to the user and the Kerberos Authentication Server (KAS).

In this section, the symbol {X,Y} indicates a message containing information (or data) X and Y. $\{X,Y\}K_z$ indicates that a message that contains the data X and Y has been enciphered using the key K_z.

Archived

Figure 22-52 shows the authentication process.

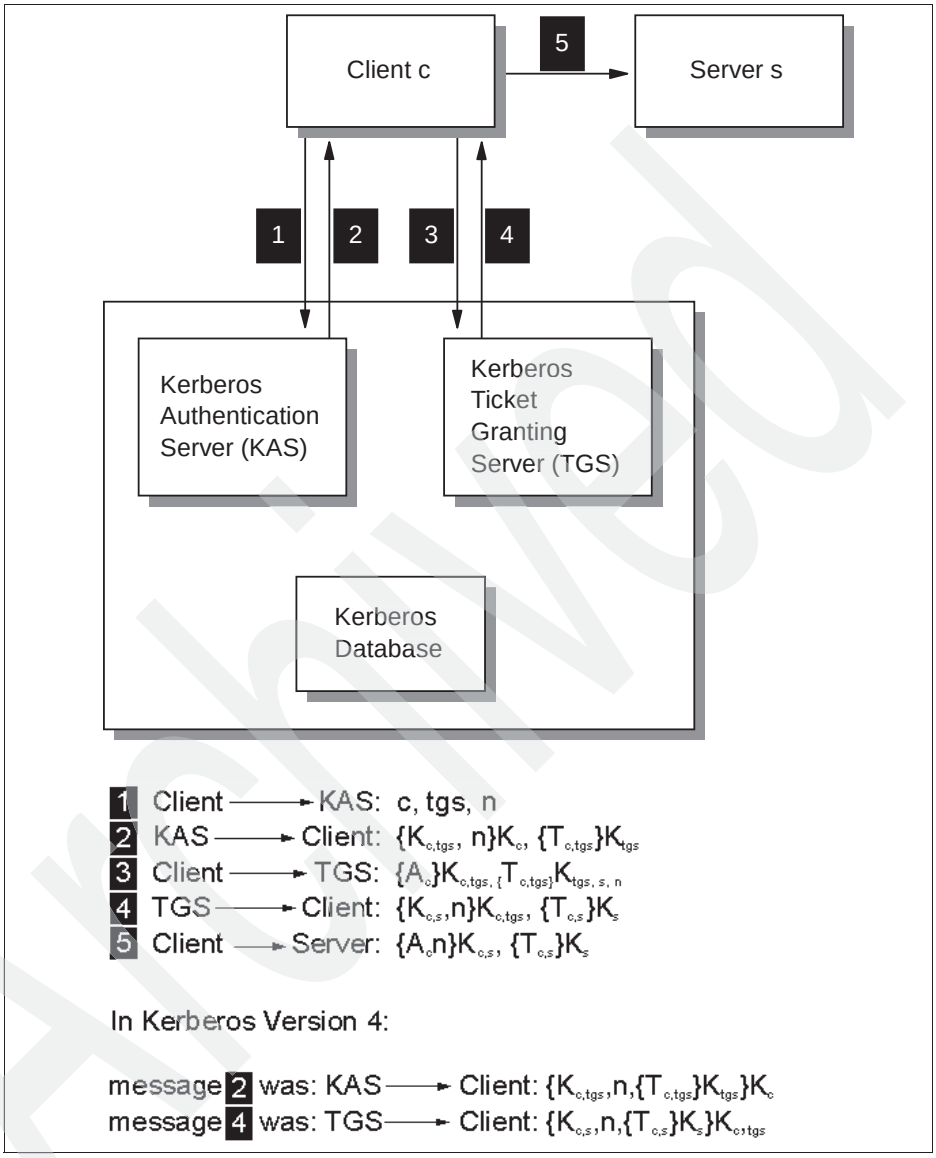


Figure 22-52 Kerberos authentication scheme

Archive

The authentication process consists of exchanging five messages (see Figure 22-52 on page 867):

1. Client → KAS

   The client sends a message {c, tgs, n} to the KAS, containing its identity (c), a nonce (a time stamp or other means to identify this request), and requests for a ticket for use with the ticket-granting server (TGS).

2. KAS → client

   The authentication server looks up the client name (c) and the service name (the ticket-granting server, tgs) in the Kerberos database and obtains an encryption key for each (K_c and K_{tgs}).

   The KAS then forms a response to send back to the client. This response contains an initial ticket $T_{c,tgs}$, which grants the client access to the requested server (the ticket-granting server). $T_{c,tgs}$ contains $K_{c,tgs}$, c, tgs, nonce, lifetime, and some other information. The KAS also generates a random encryption key $K_{c,tgs}$, called the session key. It then encrypts this ticket using the encryption key of the ticket-granting server (K_{tgs}). This produces what is called a *sealed ticket* $\{T_{c,tgs}\}K_{tgs}$. A message is then formed consisting of the sealed ticket and the TGS session key $K_{c,tgs}$.

   Note: In Kerberos Version 4, the message is:

   {Kc,tgs,n,{Tc,tgs}Ktgs}Kc

   While in Kerberos Version 5, the message is of a simpler form:

   {Kc,tgs, n}Kc, {Tc,tgs}Ktgs

   This simplifies the (unnecessary) double encryption of the ticket.

3. Client → TGS

   Upon receiving the message, the client decrypts it using its secret key K_c, which is only known to it and the KAS. It checks to see if the nonce (n) matches the specific request, and then caches the session key $K_{c,tgs}$ for future communications with the TGS.

   The client then sends a message to the TGS. This message contains the initial ticket $\{T_{c,tgs}\}K_{tgs}$, the server name (s), a nonce, and a new authenticator A_c containing a time stamp. A_c is {c, nonce}. The message is:

   $\{A_c\}K_{c,tgs}$, $\{T_{c,tgs}\}K_{tgs}$, s, n

4. TGS → client

   The ticket-granting server (TGS) receives the above message from the client (c), and first deciphers the sealed ticket using its TGS encryption key. (This ticket was originally sealed by the Kerberos authentication server in step 2 using the same key.) From the deciphered ticket, the TGS obtains the TGS-session-key. It uses this TGS session key to decipher the sealed authenticator. (Validity is checked by comparing the client name both in the ticket and in the authenticator, the TGS server name in the ticket, the network address that must be equal in the ticket, in the authenticator, and in the received message.)

   Finally, it checks the current time in the authenticator to make certain the message is recent. This requires that all the clients and servers maintain their clocks within some prescribed tolerance. The TGS now looks up the server name from the message in the Kerberos database and obtains the encryption key (K_s) for the specified service.

   The TGS forms a new random session key $K_{c,s}$ for the benefit of the client (c) and the server (s), and then creates a new ticket $T_{c,s}$ containing:

   $K_{c,s}$, n, nonce, lifetime,

   It then assembles and sends a message to the client.

   Note: In Kerberos Version 4, the message is:

   {Kc,s,n,{Tc,s}Ks}Kc,tgs

   While in Kerberos Version 5, the message is of a simpler form:

   {Kc,s,n}Kc,tgs, {Tc,s}Ks

   This simplifies the (unnecessary) double encryption of the ticket.

5. Client → server

   The client receives this message and deciphers it using the TGS session key that only it and the TGS share. From this message, it obtains a new session key $K_{c,s}$ that it shares with the server (s) and a sealed ticket that it cannot decipher because it is enciphered using the server's secret key K_s.

   The client builds an authenticator and seals it using the new session key $K_{c,s}$. At last, it sends a message containing the sealed ticket and the authenticator to the server (s) to request its service.

   The server (s) receives this message and first deciphers the sealed ticket using its encryption key, which only it and KAS know. It then uses the new session key contained in the ticket to decipher the authenticator and does the same validation process that was described in step 4.

After the server has validated a client, an option exists for the client to validate the server. This prevents an intruder from impersonating the server. The client requires then that the server sends back a message containing the time stamp (from the client's authenticator, with one added to the time stamp value). This message is enciphered using the session key that was passed from the client to the server.

Let us summarize some of the central points in this scheme:

- In order for the workstation to use any end server, a ticket is required. All tickets, other than the first ticket (also called the *initial ticket*), are obtained from the TGS. The first ticket is special; it is a ticket for the TGS itself and is obtained from the Kerberos authentication server.
- Every ticket is associated with a session key that is assigned every time a ticket is allocated.
- Tickets are reusable. Every ticket has a lifetime, typically eight hours. After a ticket has expired, you have to identify yourself to Kerberos again, entering your login name and password.
- Unlike a ticket, which can be reused, a new authenticator is required every time the client initiates a new connection with a server. The authenticator carries a time stamp within it, and the authenticator expires a few minutes after it is issued. (This is the reason why clocks must be synchronized between clients and servers.)
- A server maintains a history of previous client requests for which the time stamp in the authenticator is still valid. This way, a server can reject duplicate requests that might arise from a stolen ticket and authenticator.

22.11.4 Kerberos database management

Kerberos needs a record of each user and service in its realm and each record keeps only the needed information, as follows:

- Principal identifier (c,s)
- Private key for this principal (K_c,K_s)
- Date of expiration for this identity
- Date of the last modification in this record
- Identity of the principal that last modified this record (c,s)
- Maximum lifetime of tickets to be given to this principal (lifetime)
- Attributes (unused)
- Implementation data (not visible externally)

The private key field is enciphered using a master key so that removing the database will not cause any problem because the master key is not in it.

The entity responsible for managing this database is the Kerberos Database Manager (KDBM). There is only one KDBM in a realm, but it is possible to have more than one Kerberos Key Distribution Server (KKDS), each one having a copy of the Kerberos database. This is done to improve availability and performance so that the user can choose one in a group of KKDSs to send its request to. The KKDS performs read-only operations, leaving the actualization to the KDBM, which copies the entire database a few times a day. This is done to simplify the operation using a Kerberos protected protocol. This protocol is basically a mutual authentication between KDBM and KKDS before a file transfer operation with checkpoints and checksum.

22.11.5 Kerberos Authorization Model

The Kerberos Authentication Model permits only the service to verify the identity of the requester but it gives no information about whether the requester can use the service or not. The Kerberos Authorization Model is based on the principle that each service knows the user so that each one can maintain its own authorization information. However, the Kerberos Authentication System can be extended by information and algorithms that can be used for authorization purposes. (This is made easier in Version 5, as shown in the following section.) Kerberos can then check if a user/client is allowed to use a certain service.

Obviously, both the client and the server applications must be able to handle the Kerberos authentication process. That is, both the client and the server must be *kerberized*.

22.11.6 Kerberos Version 5 enhancements

Kerberos Version 5 has a number of enhancements over Version 4. Some of the important ones are:

- Use of encryption has been separated into distinct program modules which allows for supporting multiple encryption systems.
- Network addresses that appear in protocol messages are now tagged with a type and length field. This allows support of multiple network protocols.
- Message encoding is now described using the Abstract Syntax Notation 1 (ASN.1) syntax in accordance with ISO standards 8824 and 8825.
- The Kerberos Version 5 ticket has an expanded format to support new features (for example, the inter-realm cooperation).
- As mentioned in 22.11.2, “Naming” on page 865, the principal identifier naming has changed.
- Inter-realm support has been enhanced.

- Authorization and accounting information can now be encrypted and transmitted inside a ticket in the authorization data field. This facilitates the extension of the authentication scheme to include an authorization scheme as well.
- A binding is provided for the Generic Security Service API (GSSAPI) to the Kerberos Version 5 implementation.

22.12 Remote access authentication protocols

Remote dial-in to the corporate intranet and to the Internet has made the remote access server a very vital part of today's internetworking services. More and more mobile users are requiring access not only to central-site resources, but to information sources on the Internet. The widespread use of the Internet and the corporate intranet has fueled the growth of remote access services and devices. There is an increasing demand for a simplified connection to corporate network resources from mobile computing devices, such as a notebook computer, or a palmtop device for e-mail access.

The emergence of remote access has caused significant development work in the area of security. The AAA (triple A) security model has evolved in the industry to address the issues of remote access security. Authentication, authorization, and accounting answers the questions who, what, and when, respectively. Here we provide a brief description of each of the three As in the AAA security model:

Authentication This is the action of determining who a user (or entity) is. Authentication can take many forms. Traditional authentication uses a name and a fixed password. Most computers work this way. However, fixed passwords have limitations, mainly in the area of security. Many modern authentication mechanisms utilize one-time passwords or a challenge-response query. Authentication generally takes place when the user first logs in to a machine or requests a service of it.

Authorization This is the action of determining what a user is allowed to do. Generally, authentication precedes authorization, but again, this is not required. An authorization request might indicate that the user is not authenticated. (we do not know who they are.) In this case, it is up to the authorization agent to determine if an unauthenticated user is allowed the services in question. In current remote authentication protocols, authorization does not merely provide yes or no answers, but it can also customize the service for the particular user.

Accounting This is typically the third action after authentication and authorization. But again, neither authentication nor authorization are required. Accounting is the action of recording what a user is doing and has done.

In the distributed client/server security database model, a number of communications servers, or clients, authenticate a dial-in user's identity through a single, central database, or authentication server. The authentication server stores all information about users, their passwords, and access privileges. Distributed security provides a central location for authentication data that is more secure than scattering the user information on different devices throughout a network. A single authentication server can support hundreds of communications servers, serving up to tens of thousand of users. Communications servers can access an authentication server locally or remotely over WAN connections.

Several remote access vendors and the Internet Engineering Task Force (IETF) have been in the forefront of this remote access security effort, and the means whereby such security measures are standardized. Remote Authentication Dial In User Service (RADIUS) and Terminal Access Controller Access Control System (TACACS) are two such cooperative ventures that have evolved out of the Internet standardizing body and remote access vendors.

Remote Authentication Dial-In User Service (RADIUS) is a distributed security system developed by Livingston Enterprises. RADIUS was designed based on a previous recommendation from the IETF's Network Access Server Working Requirements Group. An IETF Working Group for RADIUS was formed in January 1996 to address the standardization of RADIUS protocol; RADIUS is now an IETF-recognized dial-in security solution (RFC 2058 and RFC 2138).

Similar to RADIUS, Terminal Access Controller Access Control System (TACACS) is an industry standard protocol specification, RFC 1492. Similar to RADIUS, TACACS receives an authentication request from an NAS client and forwards the user name and password information to a centralized security server. The centralized server can either be a TACACS database or an external security database. Extended TACACS (XTACACS) is a version of TACACS with extensions that Cisco added to the basic TACACS protocol to support advanced features. TACACS+ is another Cisco extension that allows a separate access server (the TACACS+ server) to provide independent authentication, authorization, and accounting services.

Although RADIUS and TACACS authentication servers can be set up in a variety of ways, depending on the security scheme of the network they are serving, the basic process for authenticating a user is essentially the same. Using a modem, a remote dial-in user connects to a remote access server (also called the network access server or NAS) with a built-in analog or digital modem. After the

modem connection is made, the NAS prompts the user for a name and password. The NAS then creates the so-called authentication request from the supplied data packet, which consists of information identifying the specific NAS device sending the authentication request, the port that is being used for the modem connection, and the user name and password.

For protection against eavesdropping by hackers, the NAS, acting as the RADIUS or TACACS client, encrypts the password before it sends it to the authentication server. If the primary security server cannot be reached, the security client or NAS device can route the request to an alternate server. When an authentication request is received, the authentication server validates the request and then decrypts the data packet to access the user name and password information. If the user name and password are correct, the server sends an authentication acknowledgment packet. This acknowledgement packet can include additional filters, such as information on the user's network resource requirements and authorization levels. The security server can, for instance, inform the NAS that a user needs TCP/IP or IPX using PPP, or that the user needs SLIP to connect to the network. It can include information about the specific network resource that the user is allowed to access.

To circumvent snooping in the network, the security server sends an authentication key, or signature, identifying itself to the security client. After the NAS receives this information, it enables the necessary configuration to allow the user the necessary access rights to network services and resources. If at any point in this log-in process all necessary authentication conditions are not met, the security database server sends an authentication reject message to the NAS device and the user is denied access to the network.

22.13 Extensible Authentication Protocol (EAP)

Extensible Authentication Protocol (EAP) is used for the exchange of authentication information. EAP is defined in RFC 2284 and is an extension to the Point-to-Point Protocol (PPP). EAP supports multiple authentication vehicles such as:

- Kerberos
- Public key authentication
- Key tokens
- One time passwords

EAP typically runs over the link layer and has a number of deployment solutions including:

- EAP MD5
- EAP-Tunneled TLS (EAP-TTLS)
- Lightweight EAP (LEAP)
- Protected EAP (PEAP)

When used in wireless communications, IEEE 802.1x defines how EAP is encapsulated in LAN frames. The wireless EAP solution is typically activated when a user connects to wireless access point (AP) and enters in authentication credentials. The AP verifies the identity of the user through a RADIUS server and, if the credentials are approved, access is granted to the user.

For further EAP details, refer to Chapter 23, “Port based network access control” on page 889.

22.14 Layer 2 Tunneling Protocol (L2TP)

L2TP permits the tunneling of PPP. Any protocol supported by PPP can be tunneled. This protocol extends the span of a PPP connection. Instead of beginning at the remote host and ending at a local ISP's point of presence (PoP), the *virtual PPP* link now extends from the remote host all the way back to the corporate gateway. L2TP tunneling is currently supported over IP/UDP. The specification is in RFC 2661.

L2TP is a consensus standard that came from the merging of two earlier tunneling protocols: Point-to-Point Tunneling Protocol (PPTP) and Layer 2 Forwarding (L2F, described in RFC 2341). These earlier protocols did not provide as complete a solution as the L2TP protocol; one addresses tunnels created by ISPs and the other addresses tunnels created by remote hosts. L2TP supports both host-created and ISP-created tunnels.

L2TP adds the ability to create a virtual private network where multiple protocols and privately addressed IP, IPX, and AppleTalk (AT) are allowed. In addition, L2TP gives remote users the ability to connect to a local ISP and tunnel through the Internet to a home network, avoiding long distance charges. It also provides a mechanism on which to solve the multiple box PPP multilink problem. (Calls connecting to different physical routers that are destined for the same MP bundle can be tunneled to the same endpoint where MP can be terminated for all links.)

22.14.1 Terminology

Before describing the protocol, we provide a definition of some L2TP terminology

- L2TP access concentrator (LAC)
  A device attached to one or more public switched telephone network (PSTN) or Integrated Services Digital Network (ISDN) lines capable of handling both the PPP operation and L2TP protocol. The LAC implements the media over which L2TP operates. L2TP passes the traffic to one or more L2TP servers (LNS).
- L2TP network server (LNS)
  An LNS operates on any platform that can be a PPP endstation. The LNS handles the server side of the L2TP protocol. Because L2TP relies only on the single media over which L2TP tunnels arrive, the LNS can have only a single LAN or WAN interface, yet is still able to terminate calls arriving from any PPP interfaces supported by a LAC, such as async, synchronous, ISDN, V.120, and so on.
- Network access servers (NAS)
  A device providing temporary, on demand network access to users. This access is point-to-point using PSTN or ISDN lines.
- Session (Call)
  L2TP creates a session when an end-to-end PPP connection is attempted between a dial-in user and the LNS, or when an outbound call is initiated. The datagrams for the session are sent over the tunnel between the LAC and the LNS. The LNS and LAC maintain the state information for each user attached to a LAC.
- Tunnel
  A tunnel is defined by an LNS-LAC pair. The tunnel carries PPP datagrams between the LAC and the LNS. A single tunnel can multiplex many sessions. A control connection operating over the same tunnel controls the establishment, release, and maintenance of all sessions and of the tunnel itself.
- Attribute value air (AVP)
  A uniform method of encoding message types and bodies. This method maximizes the extensibility while permitting interpretability of L2TP.

Archived

22.14.2 Protocol overview

Because the host and the gateway share the same PPP connection, they can take advantage of PPP's ability to transport protocols other than just IP. For example, L2TP tunnels can support remote LAN access as well as remote IP access. Figure 22-53 outlines a basic L2TP configuration.

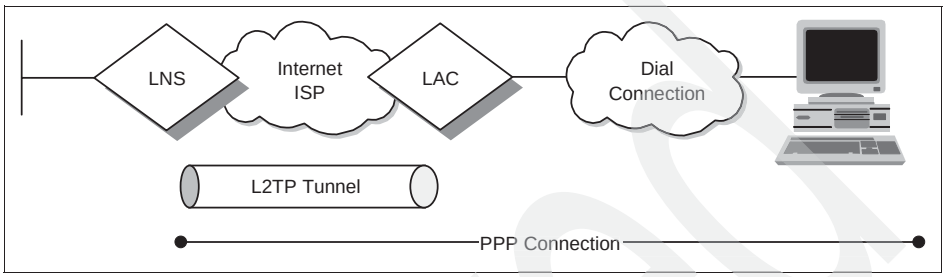


Figure 22-53 Layer 2 Tunnel Protocol (L2TP) scenario

Referring to Figure 22-53, the following actions occur:

1. The remote user initiates a PPP connection.
2. The NAS accepts the call.
3. The NAS identifies the remote user using an authorization server.
4. If the authorization is OK, the NAS/LAC initiates an L2TP tunnel to the desired LNS at the entry to the enterprise.
5. The LNS authenticates the remote user through its authentication server and accepts the tunnel.
6. The LNS confirms acceptance of the call and the L2TP tunnel.
7. The NAS logs the acceptance.
8. The LNS exchanges PPP negotiation with the remote user.
9. End-to-end data is now tunneled between the remote user and the LNS.

L2TP is actually another variation of an IP encapsulation protocol. As shown in Figure 22-54 on page 878, an L2TP tunnel is created by encapsulating an L2TP frame inside a UDP packet, which in turn is encapsulated inside an IP packet whose source and destination addresses define the tunnel's endpoints. Because the outer encapsulating protocol is IP, clearly IPSec protocols can be applied to this composite IP packet, thus protecting the data that flows within the L2TP tunnel. AH, ESP, and ISAKMP/Oakley protocols can all be applied in a straightforward way.

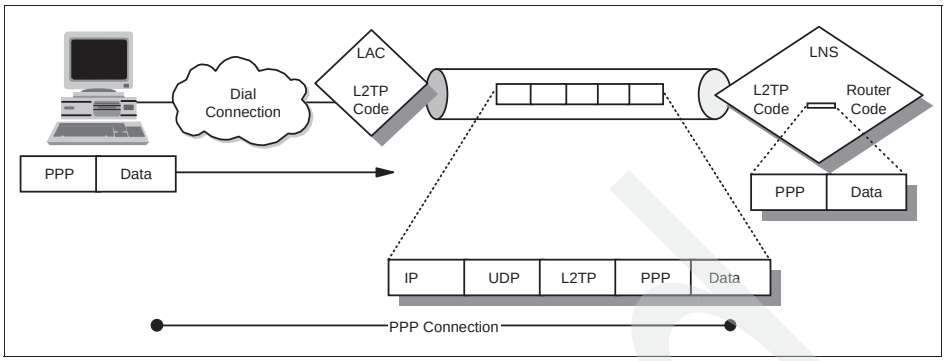


Figure 22-54 L2TP packet changes during transit

L2TP can operate over UDP/IP and support the following functions:

- Tunneling of single user dial-in clients
- Tunneling of small routers, for example, a router with a single static route to set up based on an authenticated user's profile
- Incoming calls to an LNS from a LAC
- Multiple calls per tunnel
- Proxy authentication for PAP and CHAP
- Proxy LCP
- LCP restart in the event that proxy LCP is not used at the LAC
- Tunnel endpoint authentication
- Hidden AVP for transmitting a proxy PAP password
- Tunneling using a local realm (that is, user@realm) lookup table
- Tunneling using the PPP user name lookup in the AAA subsystem (22.12, “Remote access authentication protocols” on page 872)

Archived

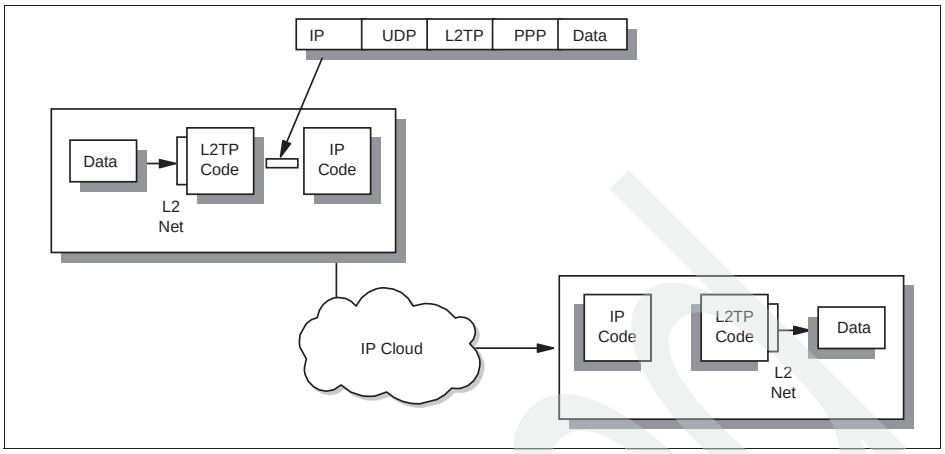


Figure 22-55 L2TP packet flow through any IP cloud

22.14.3 L2TP security issues

Although L2TP provides cost-effective access, multiprotocol transport, and remote LAN access, it does not provide cryptographically robust security features. For example:

- Authentication is provided only for the identity of tunnel endpoints, but not for each individual packet that flows inside the tunnel. This can expose the tunnel to man-in-the-middle and spoofing attacks.
- Without per-packet integrity, it is possible to mount denial-of-service attacks by generating bogus control messages that can terminate either the L2TP tunnel or the underlying PPP connection.
- L2TP itself provides no facility to encrypt user data traffic. This can lead to embarrassing exposures when data confidentiality is an issue.
- While the payload of the PPP packets can be encrypted, the PPP protocol suite does not provide mechanisms for automatic key generation or for automatic key refresh. This can lead to someone listening in on the wire to finally break that key and gain access to the data being transmitted.

Realizing these shortcomings, the PPP Extensions Working Group of the IETF considered how to remedy these shortfalls. Rather than duplicate work done elsewhere, it was decided to recommend using IPSec within L2TP. This is described in RFC 2888.

In summary, Layer 2 Tunnel Protocols are an excellent way of providing cost-effective remote access. And when used in conjunction with IPSec, they are an excellent technique for providing secure remote access. However, without

complementary use of IPSec, an L2TP tunnel alone does not furnish adequate security.

22.15 Secure Electronic Transaction (SET)

SET is the outcome of an agreement by MasterCard International and Visa International to cooperate on the creation of a single electronic credit card system. Prior to SET, each organization had proposed its own protocol and each had received support from a number of networking and computing companies. Now, most of the major players are behind the SET specification (for example, IBM, Microsoft, Netscape, and GTE).

The following sections describes at a high level the components and processes that make up the specification. Refer to the MasterCard and Visa home pages for more information about SET.

22.15.1 SET roles

The SET specification defines several roles involved in the payment process:

The merchant This is any seller of goods, services, or information.

The acquirer This is the organization that provides the credit card service and keeps the money flowing. The most widely known acquirers are MasterCard and Visa.

The issuer This is the organization that issued the card to the purchaser in the first place. Usually, this is a bank or some other financial institution, which would know the purchaser best.

The cardholder This is the Web surfer, who has been given a credit card by the issuer and now wants to exercise his or her purchasing power on the Web.

The acquirer payment gateway
This provides an interface between the merchant and the bankcard network used by the acquirer and the issuer. It is important to remember that the bankcard network already exists. The acquirer payment gateway provides a well-defined, secure interface to that established network from the Internet. Acquirer payment gateways will be operated on behalf of the acquirers, but they might be provided by third-party organizations, such as Internet service providers (ISPs).

Archive

The certificate authority SET processing uses public key cryptography, so each element of the system need one or more public key certificates. Several layers of CA are described in the specification. (We discuss SET certificates in 22.15.3, “The SET certificate scheme” on page 883.)

22.15.2 SET transactions

The SET specification describes a number of transaction flows for purchasing, authentication, payment reversal, and so on. Figure 22-56 shows the transactions involved in a typical online purchase.

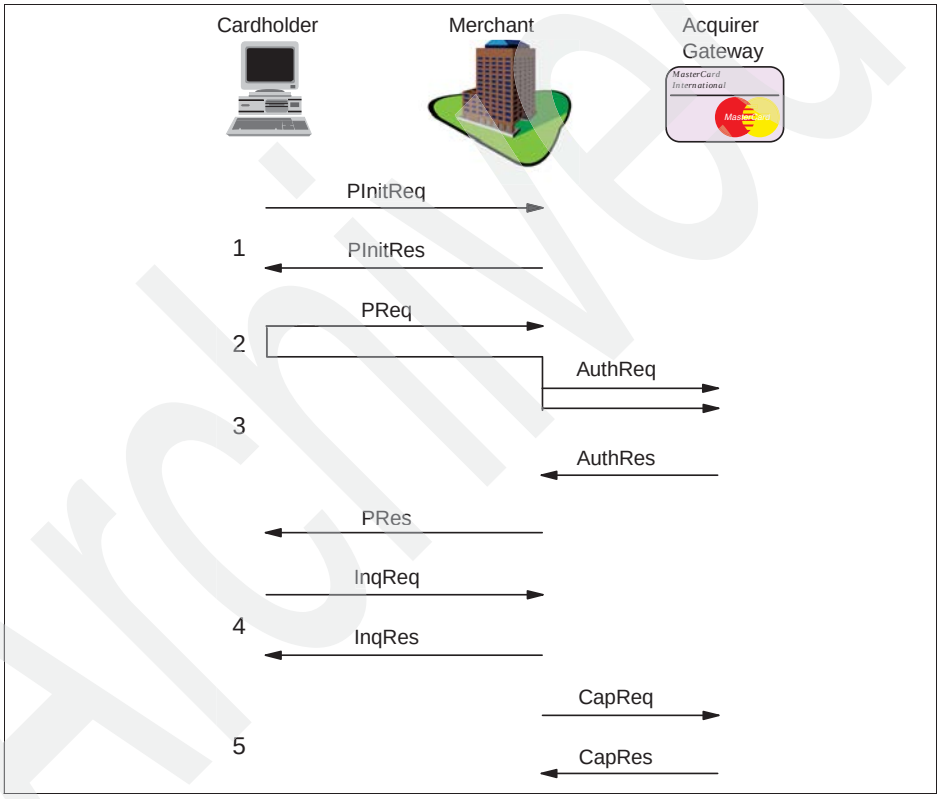


Figure 22-56 Typical SET transaction sequence

Archived

The diagram shows the following transactions (each transaction consists of a request/response pair):

1. PInit

   This initializes the system, including details such as the brand of card being used and the certificates held by the cardholder. SET does not insist that cardholders have signing certificates, but it does recommend them. A cardholder certificate binds the credit card account number to the owner of a public key. If the acquirer receives a request for a given card number signed with the cardholder's public key, it knows that the request came from the real cardholder. To be precise, it knows that the request came from a computer where the cardholder's keyring was installed and available. It *could* still be a thief who had stolen the computer and cracked the keyring password.

2. Purchase order

   This is the request from the cardholder to buy something. The request message is in fact two messages combined, the order instruction (OI), which is sent in the clear to the merchant, and the purchase instruction (PI), which the merchant passes on to the acquirer payment gateway. The PI is encrypted in the public key of the acquirer, so the merchant cannot read it. The merchant stores the message for later transmission to the acquirer. The PI also includes a hash of the OI, so the two messages can only be handled as a pair. Note that the card number is only placed in the PI portion of the request. This means that the merchant never has access to it, thereby preventing a fraudulent user from setting up a false store front to collect credit card information.

   The purchase order has a response, which is usually sent (as shown here) after acquirer approval has been granted. However, the merchant can complete the transaction with the cardholder before authorization, in which case the cardholder would see a message that the request was accepted pending authorization.

3. Authorization

   In this request, the merchant asks the acquirer, through the acquirer payment gateway, to authorize the request. The message includes a description of the purchase and the cost. It also includes the PI from the purchase order that the cardholder sent. In this way, the acquirer knows that the merchant and the cardholder both agree on what is being purchased and the amount.

   When the acquirer receives the request, it uses the existing bank card network to authorize the request and sends back an appropriate response.

4. Inquiry

   The cardholder might want to know how his or her request is proceeding. The SET specification provides an inquiry transaction for that purpose.

5. Capture

   Up to this point, no money has changed hands. The capture request from the merchant tells the acquirer to transfer the previously authorized amount to its account.

   In fact, capture can be incorporated as part of the authorization request/response (see the previous information). However, there are situations in which the merchant might want to capture the funds later. For example, most mail order operations do not debit the credit card account until the goods have been shipped.

There are several other transactions within the SET specification, but the previous summary shows the principles on which it is based.

22.15.3 The SET certificate scheme

The SET specification envisions hundreds of thousands of participants worldwide. Potentially, each of these would have at least one public key certificate. In fact, the protocol calls for an entity to have multiple certificates in some cases. For example, the acquirer payment gateways need one for signing messages and another for encryption purposes.

Archived

Key management on such a large scale requires something beyond a simple, flat certification structure. The organization of certifying authorities proposed for SET is shown in Figure 22-57.

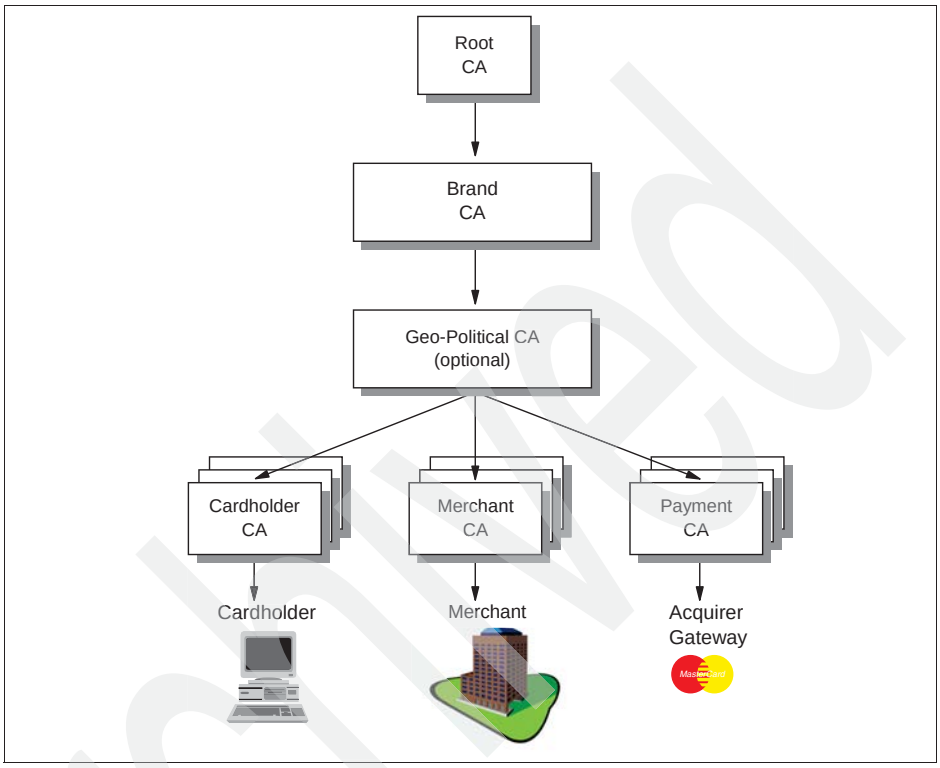


Figure 22-57 SET certifying authorities

At the top of the certificate chain, the root certifying authority is to be kept offline under extremely tight arrangements. It will only be accessed when a new credit card brand joins the SET consortium. At the next level in the hierarchy, the brand level CAs are also very secure. They are administered independently by each credit card brand.

There is some flexibility permitted under each brand for different operating policies. It would be possible to set up CAs based on region or country, for example. At the base of the CA hierarchy are the CAs that provide certificates for merchants, cardholders, and acquirer payment gateways. The SET specification provides protocols for merchants and cardholders to request certificates online. It is important to have a simple process because SET aims to encourage cardholders to have their own certificates. It envisions the cardholder surfing to the CA Web site, choosing a Request Certificate option to invoke the certificate

request application on the browser, and then filling in a form to send and receive the certificate request.

Of course, if the system allows certificates to be created easily, it must also be able to revoke them easily in the event of a theft or other security breach. The SET specification includes some certificate update and revocation protocols for this purpose. Although the mechanism for requesting a certificate might be simple, there is still a need for user education. For example, it is obvious that a cardholder needs to notify the credit card company if his or her wallet is stolen, but less obvious that he or she also needs to notify them if his or her computer is stolen. However, if the computer includes his keyring file containing the private key and certificate, it might allow the thief to go shopping at the cardholder's expense.

22.16 RFCs relevant to this chapter

The following RFCs provide detailed information about the TCP/IP security solutions presented in this chapter:

- RFC 1492 – An Access Control Protocol, Sometimes Called TACACS (July 1993)
- RFC 1579 – Firewall-Friendly FTP (February 1994)
- RFC 1928 – SOCKS Protocol Version 5 (March 1996)
- RFC 1929 – Username/Password Authentication for SOCKS V5 (March 1996)
- RFC 1961 – GSS-API Authentication Method for SOCKS Version 5 (June 1996)
- RFC 2003 – IP Encapsulation within IP (October 1996)
- RFC 2104 – HMAC: Keyed-Hashing for Message Authentication (February 1997)
- RFC 2138 – Remote Authentication Dial In User Service (RADIUS) (April 1997)
- RFC 2315 – PKCS 7: Cryptographic Message Syntax Version 1-5 (March 1998)
- RFC 2403 – The Use of HMAC-MD5-96 within ESP and AH (November 1998)
- RFC 2404 – The Use of HMAC-SHA-1-96 within ESP and AH (November 1998)

- RFC 2405 – The ESP DES-CBC Cipher Algorithm With Explicit IV (November 1998)
- RFC 2407 – The Internet IP Security Domain of Interpretation for ISAKMP (November 1998)
- RFC 2410 – The NULL Encryption Algorithm and Its Use With IPSec (November 1998)
- RFC 2411 – IP Security Document Roadmap (November 1998)
- RFC 2412 – The OAKLEY Key Determination Protocol (November 1998)
- RFC 2661 – Layer Two Tunneling Protocol “L2TP” (August 1999)
- RFC 2888 – Secure Remote Access with L2TP (August 2000)
- RFC 2986 – PKCS #10: Certification Request Syntax Specification Version 1.7 (November 2000)
- RFC 3022 – The IP Network Address Translator (NAT) (January 2001)
- RFC 3162 – Radius and IPv6 (August 2001)
- RFC 3174 – US Secure Hash Algorithm 1 (SHA1) (September 2001)
- RFC 3207 – SMTP Service Extension for Secure SMTP over Transport Layer Security (February 2002)
- RFC 3365 – Strong Security Requirements for Internet Engineering Task Force Standard Protocols (August 2002)
- RFC 3447 - Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 (February 2003)
- RFC 3514 – The Security Flag in the IPv4 Header (April 2003)
- RFC 3586 – IP Security Policy (IPSP) Requirements (August 2003)
- RFC 3686 – Using Advanced Encryption Standard (AES) Counter Mode With IPSec Encapsulating Security Payload (ESP) (January 2004)
- RFC 3711 – The Secure Real-time Transport Protocol (SRTP) (March 2004)
- RFC 3715 – IPSec-Network Address Translation (NAT) Compatibility Requirements (March 2004)
- RFC 3748 – Extensible Authentication Protocol (EAP) (June 2004)
- RFC 3749 – Transport Layer Security Protocol Compression Methods (May 2004)
- RFC 3750 – Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Certificate Handling (April 2004)
- RFC 3751 – Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification (April 2004)

- RFC 3852 – Cryptographic Message Syntax (CMS) (July 2004)
- RFC 3871 – Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure (September 2004)
- RFC 4033 – DNS Security Introduction and Requirements (March 2005)
- RFC 4050 – The Secure Shell (SSH) Protocol Assigned Numbers (April 2005)
- RFC 4051 – The Secure Shell (SSH) Protocol Architecture (April 2005)
- RFC 4052 – The Secure Shell (SSH) Authentication Protocol (April 2005)
- RFC 4053 – The Secure Shell (SSH) Transport Layer Protocol (April 2005)
- RFC 4054 – The Secure Shell (SSH) Connection Protocol (May 2005)
- RFC 4055 – Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints (June 2005)
- RFC 4056 – Generic Message Exchange Authentication for the Secure Shell Protocol (SSH) (June 2005)
- RFC 4120 – The Kerberos Network Authentication Service (V5) (July 2005)
- RFC 4301 – Security Architecture for the Internet Protocol (December 2005)
- RFC 4302 – IP Authentication Header (December 2005)
- RFC 4303 – IP Encapsulating Security Payload (ESP) (December 2005)
- RFC 4306 – Internet Key Exchange (IKEv2) Protocol (December 2005)
- RFC 4344 – The Secure Shell (SSH) Transport Layer Encryption Modes
- RFC 4346 – The Transport Layer Security (TLS) Protocol Version 1.1 (April 2006)
- RFC 4366 – Transport Layer Security (TLS) Extensions (April 2006)
- RFC 4470 – Minimally Covering NSEC Records and DNSSEC On-line Signing (April 2006)

Archived

Archived

23

Port based network access control

IP networks are often deployed without a mechanism to restrict or control access to computing resources. Unrestricted access, compounded with today's need to create well-connected networks, greatly increase the exposure of unauthorized access. This chapter provides an overview of port-based network access control (NAC) as deployed with 802.1x. 802.1x provides a framework for authentication and authorization of devices interconnected by local area networks.

© Copyright IBM Corp. 1989-2006. All rights reserved.

23.1 Port based network access control (NAC) overview

Port based network access control (NAC) is an initiative that uses the network infrastructure to authenticate and authorize devices interconnected by local area networks through the use of the IEEE standard 802.1x. With 802.1x, network administrators can control network access and apply network polices based on the outcome of the authentication process for each IEEE 802 physical port (including wired and wireless connections). 802.1x gives administrators the control to identify who is accessing computing resources and what access is permitted at the data link and network layers.

The following terminology is used with IEEE 802.1x deployments:

Authenticator An entity at one end of a point-to-point LAN segment that facilities authentication of the entity to the other end of the link.

Authentication exchange The two-party conversation between systems performing an authentication process.

Authentication process The cryptographic operations and supporting data frames that perform the actual authentication.

Authentication server An entity that provides an authentication service to an authenticator. This service determines, from the credentials provided by supplicant, whether the supplicant is authorized to access the services provided by the system in which the authenticator resides.

Authentication transport The datagram session that actively moves the authentication exchange between two systems.

Bridge port A port of an IEEE 802.1D or 802.1Q bridge.

Edge port A bridge port attached to a LAN that has no other bridges attached to it.

Network access port A point of attachment of a system to a LAN. It can be a physical port, such as a single LAN MAC attached to a physical LAN segment, or a logical port, for example, an IEEE 802.11 association between a station and an access point.

Port access entity (PAE) The protocol entity associated with a port. It can support the protocol functionality associated with the authenticator, the supplicant, or both.

Archived

Supplicant An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an authenticator attached to the other end of that link.

23.2 Port based NAC component overview

Port based network admission control specifies the framework for controlling admission to network resources by manipulating the logical state of the network port. The framework is created through the use of network components, many commonly found in local area networks, that provide additional authentication and authorization characteristics:

Port access entity The port access entity (PAE) is the logical component of the physical IEEE 802 network port.

Authenticator The authenticator is responsible for receiving device authentication requests and passing them on to the authentication server. The authenticator is the policy enforcement point. Traffic is not allowed to pass until the port state is authorized. The authenticator must support 802.1x. In a typical 802.1x installation, the edge switch acts as the autheticator. By supporting 802 medium types, this edge switch can be wired or wireless.

Supplicant The supplicant communicates its credentials to the authenticator in order to obtain access to specified LAN services. The supplicant is a small piece of software that is installed and runs on the end user's workstation or device. The protocol responsible for transmitting the authentication information to the authenticator is EAPoL, or Extensible Authentication Protocol over LANs (see "Extensible Authentication Protocol over LANs (EAPoL)" on page 894).

Authentication server As part of an 802.1x solution, the primary responsibility of the authentication server is to authenticate credentials. These credentials are those originated by the supplicant, passed to the server through the authenticator. After the credentials have been verified, the server can then provide authorization instructions back to the authenticator (which is typically the edge switch).

23.3 Port based network access control operation

Port access control provides the ability to grant or deny network access to computing resources. 802.1x uses the concept of “controlled” and “uncontrolled” network ports. These port types are a logical representation of a physical port residing on the authenticator.

Figure 23-1 depicts the concept of controlled and uncontrolled port types. The port access control operation creates two distinct access methods to the network.

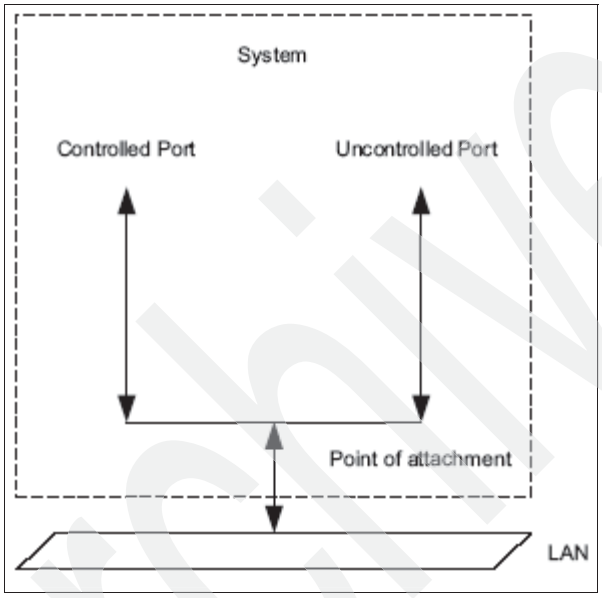


Figure 23-1 802.1x port operations

The port types are:

Uncontrolled ports Ports that are uncontrolled permit the uncontrolled exchange of frames between the authenticator and computing resources that are interconnected by the local area network.

Controlled ports Ports that are controlled permit frames between the authenticator and computing resources that are interconnected by the local area network if the port status is authenticated.

The default state of an a 802.1x controlled port is to treat all traffic as unauthorized. This forces the controlled port into an unauthorized state that

Archived

permits only the exchange of authentication traffic, or more specifically, EAPoL frames.

Figure 23-2 illustrates the operation of a controlled port in an unauthorized state.

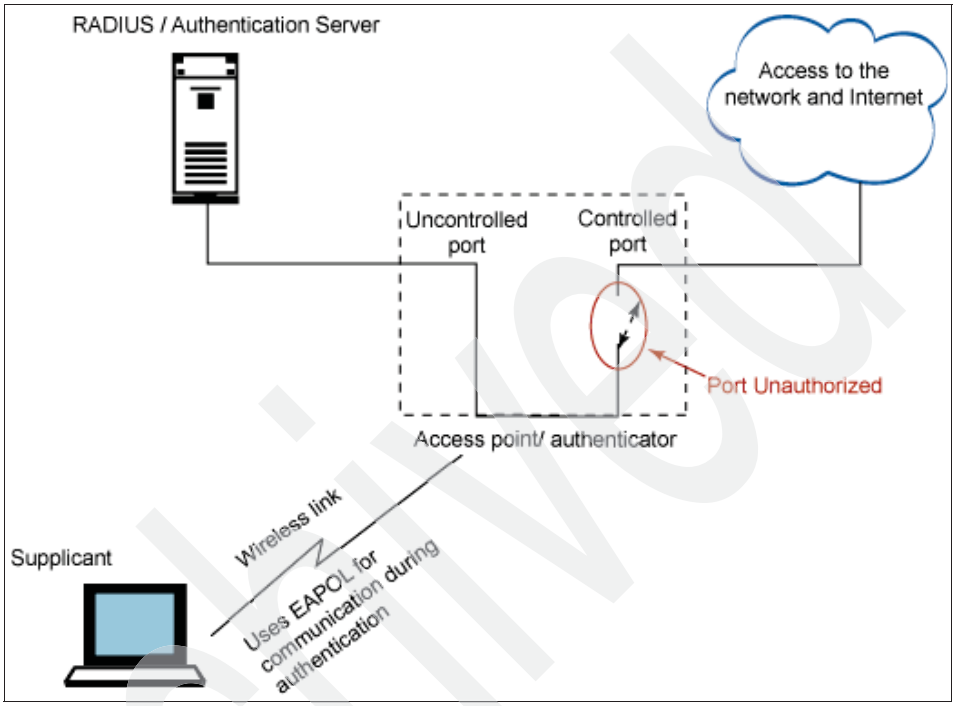


Figure 23-2 Unauthorized port

The authentication traffic that is passed represents traffic where the supplicant tries to establish its credentials through EAPoL (see “Extensible Authentication Protocol over LANs (EAPoL)” on page 894) to the authentication server using the “uncontrolled” port. The authenticator acts as a passage for the authentication traffic between the supplicant and authentication server. The authentication server verifies and validates the authenticity of the supplicant to access the network resources. The authenticator grants or rejects access for the supplicant based on the resulting inquiry from the authentication server (RADIUS server). After the supplicant authenticates successfully with the authentication server, the network port moves the controlled port to the “authorized” state, allowing the supplicant to access the computing resources.

Figure 23-3 illustrates the controlled port status after a successful authentication from a resulting EAPoL session.

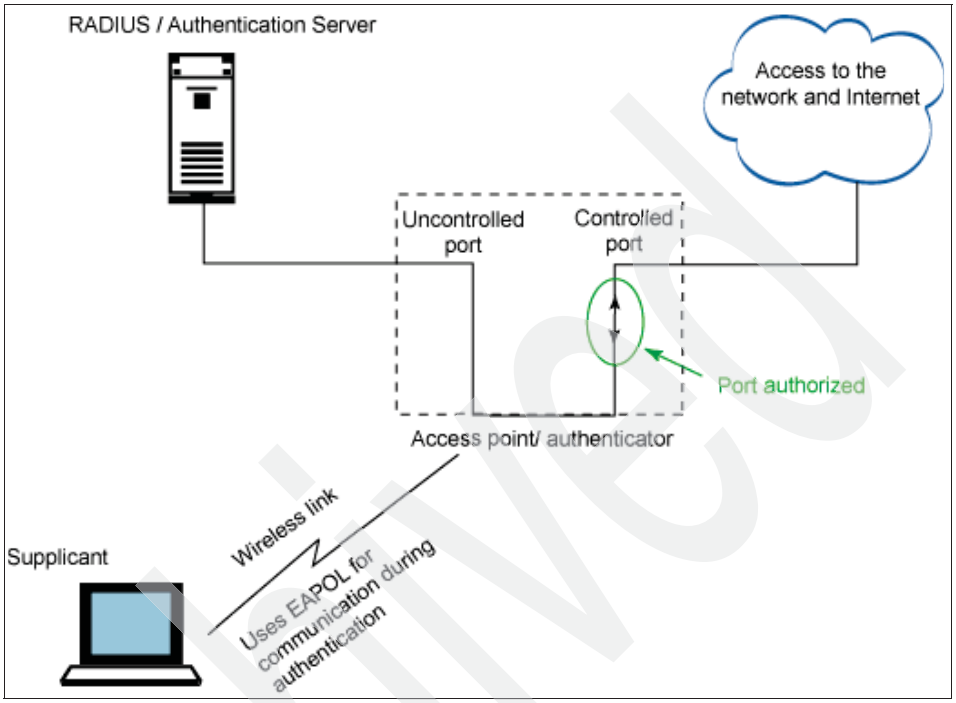


Figure 23-3 Authorized port

Extensible Authentication Protocol over LANs (EAPoL)

Extensible Authentication Protocol (EAP) is used for the exchange of authentication information between the supplicant and the authentication server. IEEE 802.1x defines an encapsulation protocol called EAP over LAN (EAPoL) to carry these EAP packets between the supplicant and the network port. The supplicant and authenticator use EAPoL for authentication and authorization communication. The authenticator then repackages these EAP packets using the RADIUS protocol and forwards them to the authentication server. The network port and authentication server use EAP over RADIUS for communication. The network port exchanges EAP authentication packets between the supplicant and authentication server with proper encapsulation, that is, EAPoL for the packets that are meant for the supplicant and RADIUS for the packets that are meant for the authentication server.

Note: EAPoL, defined in the 802.1x standard, is just an encapsulation protocol to exchange EAP authentication information between the supplicant and authenticator.

EAP is defined in RFC 3478. EAP has the ability to support multiple authentication mechanisms, making it the best candidate for passing authentication information from differing computing resources within local area networks. The different authentication mechanisms include One Time Password (OTP), Message Digest 5 (MD5), Transport Layer Security (TLS), Tunneled Transport Layer Security (TTLS), and Protected Extensible Authentication Protocol (PEAP) to provide authentication. Each mechanism has a separate RFC to describe its usage over EAP. EAP also provides the ability to support future authentication mechanisms.

Figure 23-4 illustrates the use of 802.1x with multiple authentication protocols.

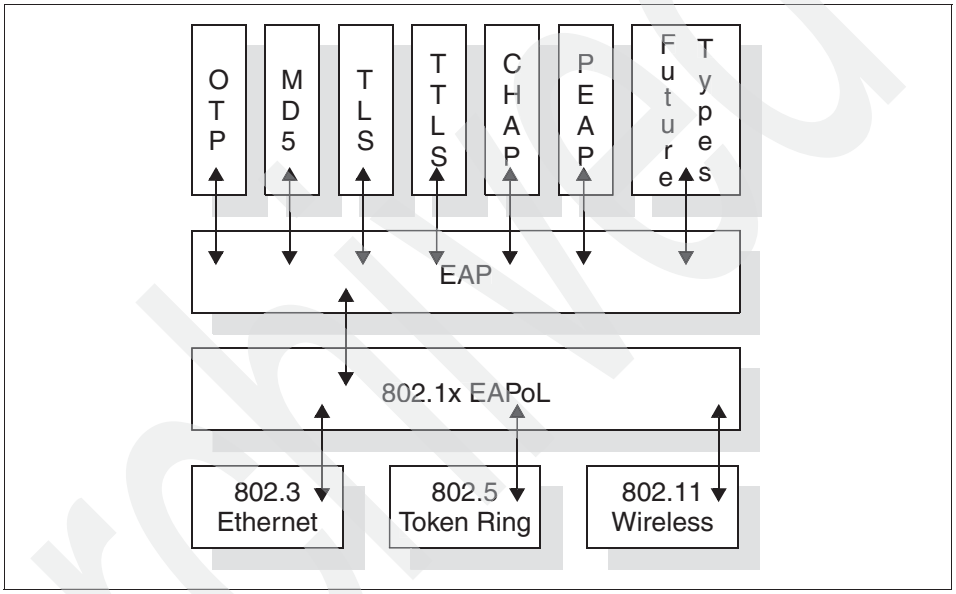


Figure 23-4 Supplicant protocol stack design

Archived

802.1x authentication process

The authentication process illustrated in Figure 23-5 depicts the basic step-by-step process of 802.1x authentication of endpoint devices (for example, workstations).

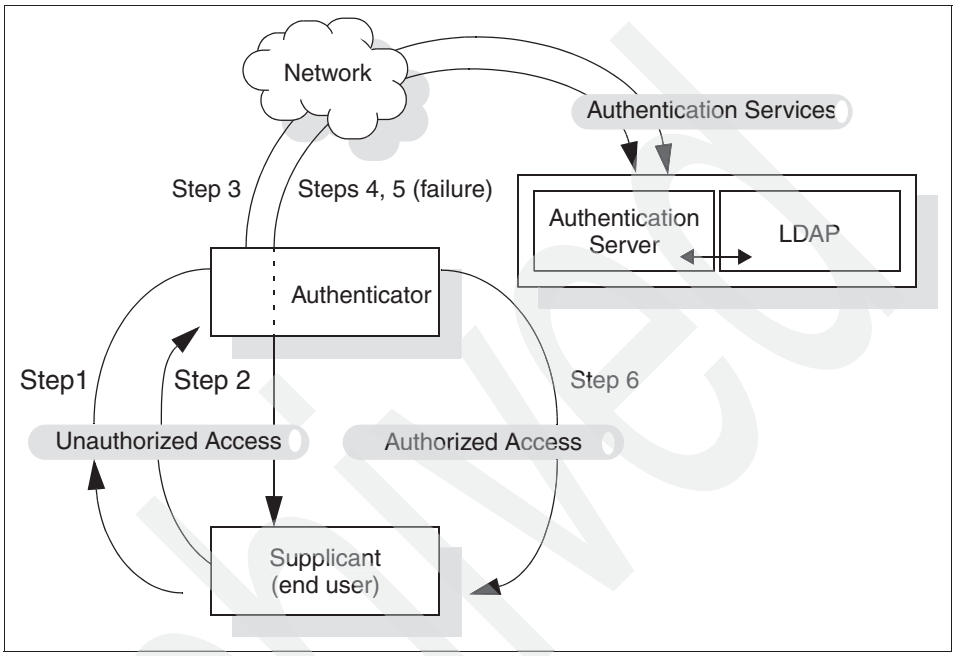


Figure 23-5 Authentication process

From the authentication process shown in Figure 23-5:

1. A workstation is attached to the network, and the supplicant initiates a session with the authenticator. The session is initiated by sending an EAPoL-Start packet. The authenticator responds by sending an EAP-Request/Identity packet to the supplicant. The network port can directly initiate the authentication process by sending an EAP-Request/Identity packet to the supplicant as soon as the port has become operable with authentication enabled on it.
2. The supplicant provides its identity by responding to the authenticator with an EAP-Response/Identity packet. The authenticator forwards this EAP packet to the authentication server over RADIUS, which verifies the supplicant's identity.

3. The authentication server sends an EAP-Request/Authentication packet to the authenticator over RADIUS and forwards this to the supplicant over EAPoL. This packet requests the supplicant to prove its credentials using the authentication type supported on the authentication server.
4. If the supplicant does not support the authentication type mentioned in the EAP-Request/Authentication packet, it responds with an EAP-Nak message, which indicates that the authentication type is not supported. The packet can also include the desired type of authentication that the supplicant supports. If the supplicant supports the authentication type, it responds with the EAP-Response/Authentication packet to the authenticator, which forwards this packet to the authentication server. The number of request and response authentication messages exchanged depends on the authentication type in use.
5. With a series of EAP-Request and -Response authentication packet exchanges, the authentication server verifies the supplicant's credentials. If credentials are proper, the authentication server sends an EAP-Success packet to the authenticator, which is then forwarded to the supplicant. The supplicant receives an EAP-Failure packet if it is not able to prove its credentials. If the authentication is successful, the authenticator moves the controlled port to the “authorized” state, allowing the supplicant to access the network and Internet.
6. If the type of authentication is successful, the supplicant receives an EAPOL-Key packet from the authenticator. IEEE 802.1x provides a way of exchanging information related to the encryption key between the supplicant and authenticator with the EAPOL-Key packet, helping in sharing a common encryption key for that particular session. If you use PEAP, TLS, or TTLS over EAP to provide authentication, the authentication server first sends information related to encryption key to the authenticator. The authenticator forwards this information as an EAPOL-Key packet to the supplicant.
7. If the supplicant wants to disconnect from the network after successful authentication, it sends an EAPOL-Logoff packet to the authenticator. The authenticator immediately moves the controlled port to the unauthorized state disabling the supplicant from accessing the network.

Figure 23-6 and Figure 23-7 show sniffer traces for two types of authentication, EAP-MD5 and PEAP, respectively. These figures depict all the packets that are exchanged during successful authentication of a supplicant.

Source	Destination	Protocol	Info
Cisco_41:6d:be	Cisco_b3:f4:71	EAPOL	Start
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, Identity [RFC3748]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, Identity [RFC3748]
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, MD5-Challenge [RFC3748]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, MD5-Challenge [RFC3748]
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Success

Figure 23-6 EAP-MD5 successful authentication

Source	Destination	Protocol	Info
Cisco_41:6d:be	Cisco_b3:f4:71	EAPOL	Start
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, Identity [RFC3748]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, Identity [RFC3748]
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, PEAP [Palekar]
Cisco_41:6d:be	Cisco_b3:f4:71	TLS	Client Hello
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, PEAP [Palekar]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, PEAP [Palekar]
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Server Hello, Certificate, Server Hello Done
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Change Cipher Spec, Encrypted Handshake Message
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, PEAP [Palekar]
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Application Data, Application Data
Cisco_41:6d:be	Cisco_b3:f4:71	TLS	Application Data, Application Data
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Application Data, Application Data
Cisco_41:6d:be	Cisco_b3:f4:71	TLS	Application Data, Application Data
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Application Data, Application Data
Cisco_41:6d:be	Cisco_b3:f4:71	TLS	Application Data, Application Data
Cisco_b3:f4:71	Cisco_41:6d:be	TLS	Application Data, Application Data
Cisco_41:6d:be	Cisco_b3:f4:71	TLS	Application Data, Application Data
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Success
Cisco_b3:f4:71	Cisco_41:6d:be	EAPOL	Key
Cisco_b3:f4:71	Cisco_41:6d:be	EAPOL	Key

Figure 23-7 PEAP successful authentication

Figure 23-8 shows a failure.

Source	Destination	Protocol	Info
Cisco_41:6d:be	Cisco_b3:f4:71	EAPOL	Start
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, Identity [RFC3748]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, Identity [RFC3748]
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Request, MD5-Challenge [RFC3748]
Cisco_41:6d:be	Cisco_b3:f4:71	EAP	Response, MD5-Challenge [RFC3748]
Cisco_b3:f4:71	Cisco_41:6d:be	EAP	Failure

Figure 23-8 EAP failure

In the 802.1x/EAPoL header (Figure 23-9 on page 899), the Protocol version field indicates that the EAPoL protocol version is 1. The Packet type field indicates the type of packet. A value of 0 for this field indicates that it is an EAP packet. The Packet body length field contains the length of the packet, excluding the Ethernet and EAPoL headers.

In the EAP header, the Code field indicates the type of EAP packet. There are four types of EAP packets that are exchanged during authentication: request,

response, success, and failure packets. The Identifier field matches the responses with requests. When a response is sent for a particular request, the value of the Identifier field in the response packet is the same as that of the request. The Length field indicates the length of the EAP packet, which includes only an EAP header and data field.

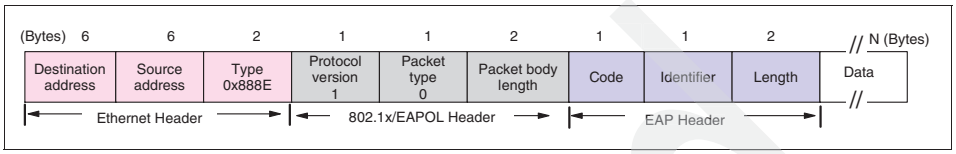


Figure 23-9 EAP packet format

EAP-Request and EAP-Response packets

In the EAP header (Figure 23-10), the Code field indicates the type of EAP packet. A value of 1 indicates that it is an EAP-Request packet. A value of 2 indicates that it is an EAP-Response packet. An additional Type field is present in the EAP header of the request or response packet. This indicates the type of request or response packet.

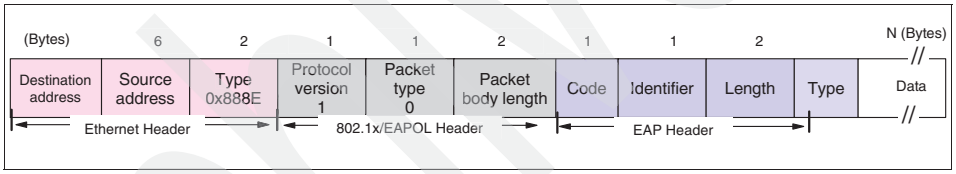


Figure 23-10 EAP-Request and EAP-Response packet format

Table 23-1 provides some major request or response types and their values. The first three are special types and the remaining types define various authentication methods.

Table 23-1 Major request and response types

Request and response type	Value
Identity	1
Notification	2
Nak (response only)	3
MD-5 challenge	4
One-time password (OTP)	5
Generic token card (GTC)	6

Request and response type	Value
TLS	13
TTLS	21
PEAP	25
MS-CHAP-V2	29

In Figure 23-11, the Type field in the EAPOL header (802.1x authentication) is 0, indicating that it is an EAP packet. In the EAP header, a value of 2 for the Code field indicates that it is a response packet. A value of 1 for the Type field indicates an Identity packet. All these values clearly give us an idea that it is an EAP-Response/Identity packet.

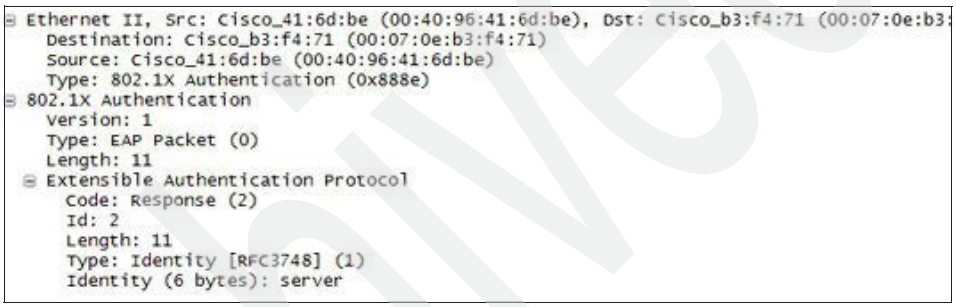


Figure 23-11 EAP-Response and Identity packet

Figure 23-12 shows a EAP-Nak packet. This packet is sent only as an EAP response to indicate that the authentication type sent in the Request packet is unacceptable to the peer. The value of the Type field in the EAP header is 3 to indicate that it is an EAP-Nak packet.

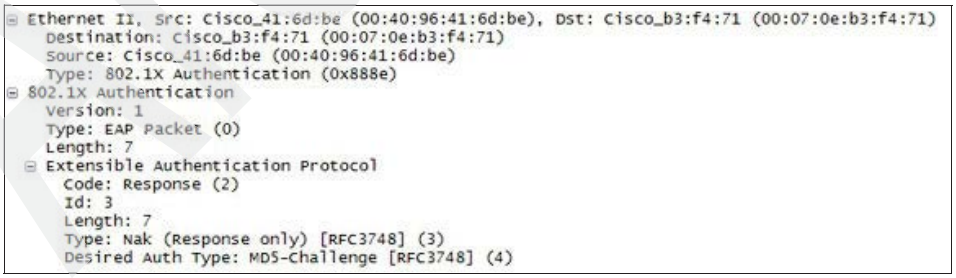


Figure 23-12 EAP Nak

Figure 23-13 shows the EAP Request/Authentication or EAP Request/PEAP packet. A value of 1 in the Code field of the EAP header indicates that it is an EAP-Request packet. A value of 25 in the Type field of the EAP header indicates that it is an EAP-Request/PEAP packet.

```
⊟ Ethernet II, Src: Cisco_b3:f4:71 (00:07:0e:b3:f4:71), Dst: Cisco_41:6d:be (00:40:96:41:6d:be)
    Destination: Cisco_41:6d:be (00:40:96:41:6d:be)
    Source: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Type: 802.1X Authentication (0x888e)
⊟ 802.1X Authentication
    Version: 1
    Type: EAP Packet (0)
    Length: 1034
  ⊟ Extensible Authentication Protocol
      Code: Request (1)
      Id: 4
      Length: 1034
      Type: PEAP [Palekar] (25)
      Flags(0xc0): Length More
      PEAP version 0
      Length: 1877
    ⊞ [EAP-TLS Fragments (1877 bytes): #8(1024), #10(853)]
    ⊞ Secure Socket Layer
```

Figure 23-13 EAP-Request/PEAP packet

EAP success packet

Figure 23-14 depicts the format of an EAP-Success packet.

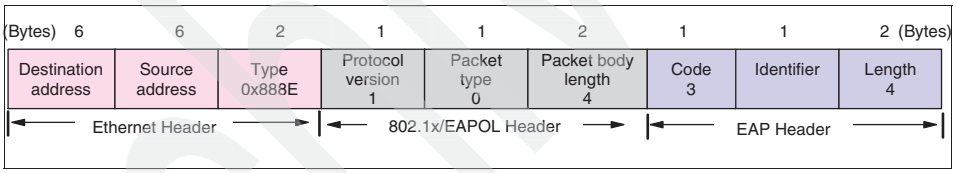


Figure 23-14 EAP-Success packet format

Figure 23-15 on page 902 shows the sniffer trace for the EAP-Success packet. The value of the Packet type field in the 802.1x header is 0, which indicates that it is an EAP packet. A value of 3 for the Code field in the EAP header indicates that it is an EAP-Success packet. Note that there is not a Data field for this packet. Therefore, the Packet body length value in the 802.1x or EAPOL header is 4 excluding the Ethernet and 802.1x headers. Similarly, the value of the Length field in the EAP header is 4, which just includes the EAP header.

Archived

```
⊟ Ethernet II, Src: Cisco_b3:f4:71 (00:07:0e:b3:f4:71), Dst: Cisco_41:6d:be (00:40:96:41:6d:be)
    Destination: Cisco_41:6d:be (00:40:96:41:6d:be)
    Source: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Type: 802.1X Authentication (0x888e)
    Trailer: 000000000000000000000000000000000000000000000000...
⊟ 802.1X Authentication
    Version: 1
    Type: EAP Packet (0)
    Length: 4
  ⊟ Extensible Authentication Protocol
      Code: Success (3)
      Id: 9
      Length: 4
```

Figure 23-15 EAP-Success packet

EAP-Failure packet

Figure 23-16 shows the complete format for the EAP-Failure packet. The Code field in the EAP header is 4 for the EAP-Failure packet. All the remaining fields are the same as that of the EAP-Success packet.

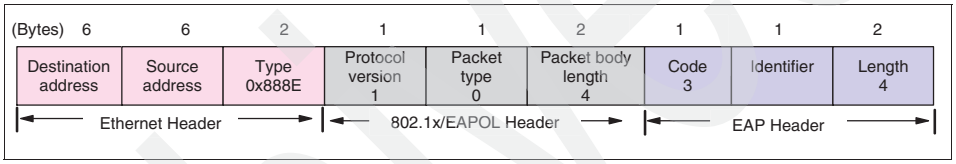


Figure 23-16 EAP-Failure packet format

Figure 23-17 shows the sniffer trace for an EAP-Failure packet.

```
⊟ Ethernet II, Src: Cisco_b3:f4:71 (00:07:0e:b3:f4:71), Dst: Cisco_41:6d:be (00:40:96:41:6d:be)
    Destination: Cisco_41:6d:be (00:40:96:41:6d:be)
    Source: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Type: 802.1X Authentication (0x888e)
    Trailer: 000000000000000000000000000000000000000000000000...
⊟ 802.1X Authentication
    Version: 1
    Type: EAP Packet (0)
    Length: 4
  ⊟ Extensible Authentication Protocol
      Code: Failure (4)
      Id: 3
      Length: 4
```

Figure 23-17 EAP-Failure packet

Archived

Figure 23-18 shows the EAPOL-Start packet format. A value of 1 for the Packet type field in the 802.1x and EAPOL header indicates that it is an EAPOL-Start packet. There is not a packet body for this packet; therefore, the value of the Packet body length field is 0.

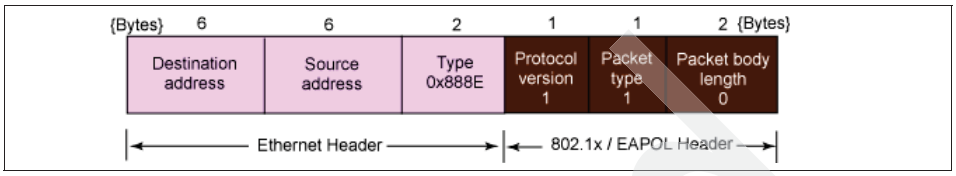


Figure 23-18 EAPoL-Start packet format

Figure 23-19 shows the sniffer trace for an EAPoL-Start packet.

```
Ethernet II, Src: Cisco_41:6d:be (00:40:96:41:6d:be), Dst: Cisco_b3:f4:71 (00:07:0e:b3:f4:71
    Destination: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Source: Cisco_41:6d:be (00:40:96:41:6d:be)
    Type: 802.1X Authentication (0x888e)
802.1X Authentication
    Version: 1
    Type: Start (1)
    Length: 0
```

Figure 23-19 EAPoL-Start packet

Figure 23-20 shows the packet format and Ethereal sniffer trace for the EAPOL-Logoff packet, respectively. A value of 2 for the Packet type field in the 802.1x/EAPOL header indicates that it is an EAPOL-Logoff packet. Because there is no pocket body for this packet, the Packet body length field is 0.

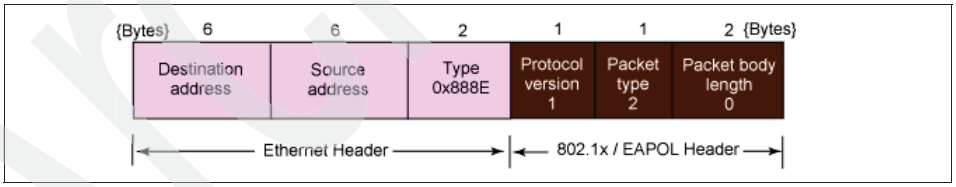


Figure 23-20 EAPoL-Logoff packet format

Archived

Figure 23-21 shows the sniffer trace for an EAPoL-Logoff packet.

```
Ethernet II, Src: Cisco_41:6d:be (00:40:96:41:6d:be), Dst: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Destination: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Source: Cisco_41:6d:be (00:40:96:41:6d:be)
    Type: 802.1X Authentication (0x888e)
802.1X Authentication
    Version: 1
    Type: Logoff (2)
    Length: 0
```

Figure 23-21 EAPoL-Logoff packet

Figure 23-22 shows the EAPOL-Key packet format. The Packet type field in the 802.1x header has the value of 3 to indicate that it is an EAPOL-Key packet.

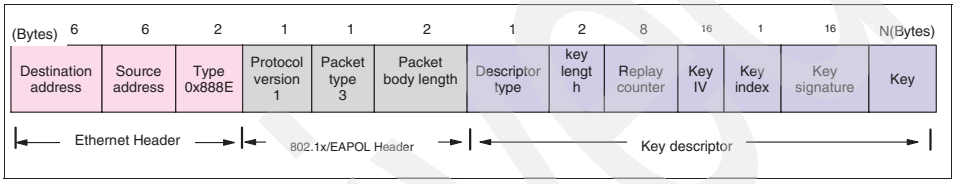


Figure 23-22 EAPoL-Key packet format

Figure 23-23 shows the sniffer trace for an EAPoL-Key packet.

```
Ethernet II, Src: Cisco_b3:f4:71 (00:07:0e:b3:f4:71), Dst: Cisco_41:6d:be (00:40:96:41:6d:be)
    Destination: Cisco_41:6d:be (00:40:96:41:6d:be)
    Source: Cisco_b3:f4:71 (00:07:0e:b3:f4:71)
    Type: 802.1X Authentication (0x888e)
802.1X Authentication
    Version: 1
    Type: Key (3)
    Length: 57
    Descriptor Type: RC4 Descriptor (1)
    Key Length: 13
    Replay Counter: 47904259776581
    Key IV: 94F18F4A118B4EA46E58E51C5A269B5C
    Key Index: broadcast, Index 2
    0... .... = Key Type: Broadcast
    .000 0010 = Index Number: 2
    Key Signature: 01D3FFE61CB7C133E8FE24980CECD7E5
    Key: 91612E4D6032D5A641C7AD0939
```

Figure 23-23 EAPoL-Key packet

23.3.1 Port based network access control functional considerations

The standard for port based network access control allows for the flexibility in deployment. Any device capable of supporting 802.1x can act as any combination of a supplicant, authenticator, and authentication server. However, to leverage the greatest amount of functionality, the IEEE 802.1x standard suggests the use of the following aspects.

Edge authentication

The IEEE 802.1x standard suggests deploying 802.1x on edge switches closest to the device needing access to computing resources, as depicted in Figure 23-5 on page 896. This approach creates the following advantages:

- Security
  All authenticated end states on the local access bridge are protected from non-authenticated end stations. If authentication was performed on a core bridge, it is possible for a malicious end station to attack authenticated end stations connected to the same local access bridge, or any number of other local access bridges between this bridge and the core bridge. These attacks are eliminated by limiting service to non-authenticated end stations directly on the local access bridge.
- Complexity
  If authentication is performed in the core of the LAN, there is the possibility of multiple bridges on a shared segment initiating authentication. To avoid this, the bridge protocol entity must manipulate the spanning tree states to make sure that only the bridge that lies in the forwarding path initiates authentication.
- Scalability
  Implementing authentication in the core of the LAN requires authentication to depend on individual MAC addresses, not just on a physical point of attachment. This, in turn, requires that the authentication state be associated with the filtering database entry for that MAC address. This increases the implementation costs and requires changes to the operation of address aging and learning. Topology changes and spanning tree reconfigureation complicate the interaction in a large network.
- Availability
  Bridged networks are frequently designed with availability as one of the primary goals. The core of the network is redundant fault-tolerant. If authentication is performed in the core, it requires reauthentication of all the end stations whenever topology changes cause port state changes in the spanning tree.
- Translational bridging
  Performing authentication at the access bridge avoids complications arising from translational bridging or VLANS. If only a single link exists between the end station and the bridge, frames do not need to be translated or tagged during the authentication exchange. The path to a core bridge might involve a variety of link types (FDDI, token ring, and so on) and packet formats (for example, VLAN tagged frames, MAC encapsulations). Were authentication to be allowed on core bridges, additional rules might be necessary in order to specify how the authentication protocols are translated.

- Multicast propagation
  If authentication occurs in the core of the LAN, it is necessary for local access bridges to forward authenciation traffic toward the core so that the authenticator can respond. Because core bridges are not able to sense the end-station connection to the local access bride port, initiation of authentication can occur either on receiving traffic from a new end station or through end-station inititation. Requiring a core bridge to maintain authentication state for each end station does not scale. In order for end-station initiation to reach the core bride, this requires that these (multicast) frames be flooded by authenciation-unaware access bridges, impatient other end stations. In contrast, if authentication occurs only on local access bridges, these multicast frames are not forwarded.

23.4 RFCs relevant to this chapter

The following RFCs provide detailed information about network admission control as presented throughout this chapter:

- RFC 2716 – PPP EAP TLS Authentication Protocol (October 1999)
- RFC 2865 – Remote Authentication Dial in User Services (RADIUS) (June 2000)
- RFC 3748 – Extensible Authentication Protocol (EAP) (June 2004)
- RFC 4017 – Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs (March 2005)
- Internet-Drafts Database Interface – EAP over UDP (EAoUDP)

Archived

24

Availability, scalability, and load balancing

This chapter discusses the various availability, scalability, and load balancing techniques used within enterprises in an attempt to ensure continuous data flow and minimize outages.

This chapter describes the following topics:

- Availability
- Scalability
- Load balancing
- Clustering
- Virtualization
- Virtual Router Redundancy Protocol (VRRP)
- Round-robin DNS
- Alternate solutions to load balancing

© Copyright IBM Corp. 1989-2006. All rights reserved.

The Internet business has grown so rapidly that continuous availability of mission-critical data and applications residing on servers is a very important requirement for enterprises. The Internet challenges companies to develop new strategies for increasing revenue and providing detailed product and delivery information. The result raises client and business partner satisfaction. Also, the management of enterprises' internal business processes that are accessed through the Internet by the workforce must be better optimized.

Increasing demands from client, business partners, and employees for access to applications and data are challenges for the development of new server and networking strategies and services. Consider the following three main aspects:

- How can availability to an enterprise's information be made available 24 hours a day and 7 days a week?
- How can these services also be guaranteed even if the number of transactions increases very rapidly, for example, because of a spike in client or business partner inquiries?
- How can the access to server applications and data be shared among parallel installed servers?

The answers are availability, scalability, and load balancing. In this chapter, we discuss techniques that can be employed to achieve availability, scalability, and load balancing. We discuss each technique at a fairly high level.

Archived

24.1 Availability

Application instances, network interfaces, and machines can fail (planned for maintenance or unplanned due to application or system error). In these cases, users must not lose their service. Recovering from application instance failure is fairly straightforward in that the application is simply restarted. Network interface failures can also be tolerated by making use of a virtual IP address, which is not tied to any particular physical interface and thus will never fail.

A *virtual IP address* can be given to a device that has one or more network interfaces. This allows the users' machines to pick up a specific IP associated with a specific machine or device. However, the IP address given is not tied to the physical IP address of the device's interfaces. Therefore, if one of the interfaces fails, the users are unaware of the failure.

Machine failure, however, is a bit more complex. Users must be able to immediately reconnect to the service without knowing that they now are using an alternate image of the application on another system. Users also must not be aware that the path to the other system has been automatically changed. The use of virtualization can be very advantageous with regards to increasing the availability of a system. Virtualization is discussed further later.

24.2 Scalability

Scalability means to provide a solution for a growing business that requires additional system capacity. When workload capacity becomes smaller due to many more new connection requests from clients or business partners, a nondisruptive growth of the current system environment must be made available.

In a traditional single system environment (no clustered systems), a nondisruptive upgrade of systems is relatively limited. In order to raise capacity, these systems have to be taken down to install new features. Therefore, they are not available for a certain time.

The implementation of clustered systems is a better approach (discussed in more depth later). Adding a new system to the cluster running equal applications instances does not impact the other systems in the cluster. This solution adds seamless capacity for a growing business. Compared to traditional systems, the user is not bound to a given system in a clustered server environment. Therefore, the management of user connections to servers is more flexible. When a new system comes online, new connections are directed to that machine taking over a new workload.

Generic techniques to enhance scalability include clustering, virtualization, and the monitoring of devices to ensure that if certain resource thresholds are met, the resources are upgraded. We discuss clustering and virtualization in more detail later.

24.3 Load balancing

Assigning applications with user connections to a specific system can overload this system's capacity, while other systems with fewer connection requests to other applications might waste free capacity.

To reach the goal for an equal level of load of all systems, these systems must be organized in a clustered system group. All systems in this cluster can provide information about their workload to the load balancing device. This device will now be responsible for distributing connection requests from users to the systems of the application servers, based on workload information.

Users are not aware of such clusters. They try to connect to a service, assuming it is running in the machine of the load balancer. The load balancer forwards the connection request to the real service provider based on the current workload of all systems in the cluster. The information about the state of the workload can be provided by a function, such as a workload manager residing in every target system.

If there is no workload information from target systems, the load balancer can use distribution rules, such as:

- A simple round-robin distribution
- Number of distributed connections

We discuss techniques used to assist with or provide load balancing, scalability, and availability next.

24.4 Clustering

In order to provide the referenced availability requirement, another system organization has to be applied. This leads to running multiple application instances on multiple machines, including TCP/IP stacks with parallel connections to the TCP/IP network. This solution, called the clustering technique in general terms, is used for load balancing purposes but is also valid for solving high availability requirements.

The clustering technique dispatches connections to target servers, excluding failed servers, from a list of target servers that can receive connections. In this way, the dispatching function avoids routing connections to a server that is not capable of satisfying such a request.

The clustering technique requires the implementation of equal application instances running on different machines. If the application, the operating system with TCP/IP stack, or the machine fails, the dispatching technique immediately provides a backup.

A user requesting service from a particular server would no longer address an application in a particular server but now would address a group of servers. The connection request is now sent to the dispatcher, who decides to which available application server it is forwarded. Therefore, users are not aware to which application server (within the group) they are connected.

The clustering technique requires addresses that refer to groups of applications. This can be solved through virtual IP addresses. A virtual IP address (VIPA) is the IP address of a group of application servers, for example, a Telnet server. This VIPA is used for a connection request. The dispatcher is the receiver of the connection request from the user. It selects from a list of available servers a real server and forwards the request to this server.

The process of selecting an available application server may be extended by the dispatcher by using different kind distribution rules. The distribution of connection requests will be discussed in the load balancing section.

Another aspect of availability to consider is when the dispatcher fails. In this case, a backup dispatcher has to be implemented with the same IP address so that users can send their connection requests to the backup dispatcher. A backup dispatcher also propagates its IP address to the network. Therefore, routers use the new path that directs the user's connection requests to the backup dispatcher.

If dispatchers maintain client/server connections, the backup dispatcher has to take over the currently running connections. A takeback process must be implemented to return running connections to the primary dispatcher.

Virtualization is also a technique used to provide availability and scalability. Virtualization has similarities to the clustering technique with regards to transparency shown to the users regarding which physical machine is being used as well as the there being no impact to the users if a machine is to fail. We discuss virtualization next.

24.5 Virtualization

Virtualization is the logical representation of resources not inhibited by physical boundaries. The main objective of virtualization is to simplify the IT infrastructure. It simplifies access to resources and the management of those resources. A user accesses the required service through standard interfaces supported and maintained by the virtualized resource. The standard interfaces allows availability issues to be minimized when changes to the IT infrastructure occur.

There many types of virtualization, and we describe some in the following sections.

For additional information, refer to:

- The IBM developerWorks® article “Virtualization in a nutshell: A pattern point of view”

  `http://www.ibm.com/developerworks/grid/library/gr-virt/`

- The IBM Redpaper *Virtualization and the On Demand Business*, REDP-9115

  `http://www.redbooks.ibm.com/redpapers/pdfs/redp9115.pdf`

Server virtualization

Many applications cannot be hosted on the same physical server due to resource conflicts. This creates issues regarding the number of servers deployed as well as the utilization of the existing resources. This lack of utilization is expensive, especially considering the cost of wasted storage space, server processing ability, and network utilization. Server virtualization is one way to resolve these issues.

Server virtualization is used to detach the applications from the physical configurations and limitations. Server virtualization is generally used as an IT optimization technique and has numerous benefits regarding availability and scalability.

Server virtualization provides the flexibility to dynamically change the allocation of system resources for the virtualized environments. The virtual servers can run on any of the physical machines. This means that the machine resources are fully shared. This makes it possible to run the physical server at high utilization levels. In addition, if any of the underlying physical resources need to be changed, it does not affect the virtualized servers. This enhances the level of scalability and availability associated with each virtual server.

Another aspect of availability to consider is if one of the virtual server instances fails. In such a case, it does not affect any of the other virtual servers currently residing on the same physical machine. Each instance of the virtual servers is

completely isolated from each other. This also eliminates any security issues or concerns regarding data leakage. Each instance of the server is also kept as a file, which can then easily be copied onto a new virtual server if the instance fails. This assists with the time taken to recover the virtual servers and the overall availability of the service provided.

As shown in Figure 24-1, many virtual servers are running off of one physical server. This also illustrates how the users are unaware that the server being used is a virtual one as opposed to a physical one.

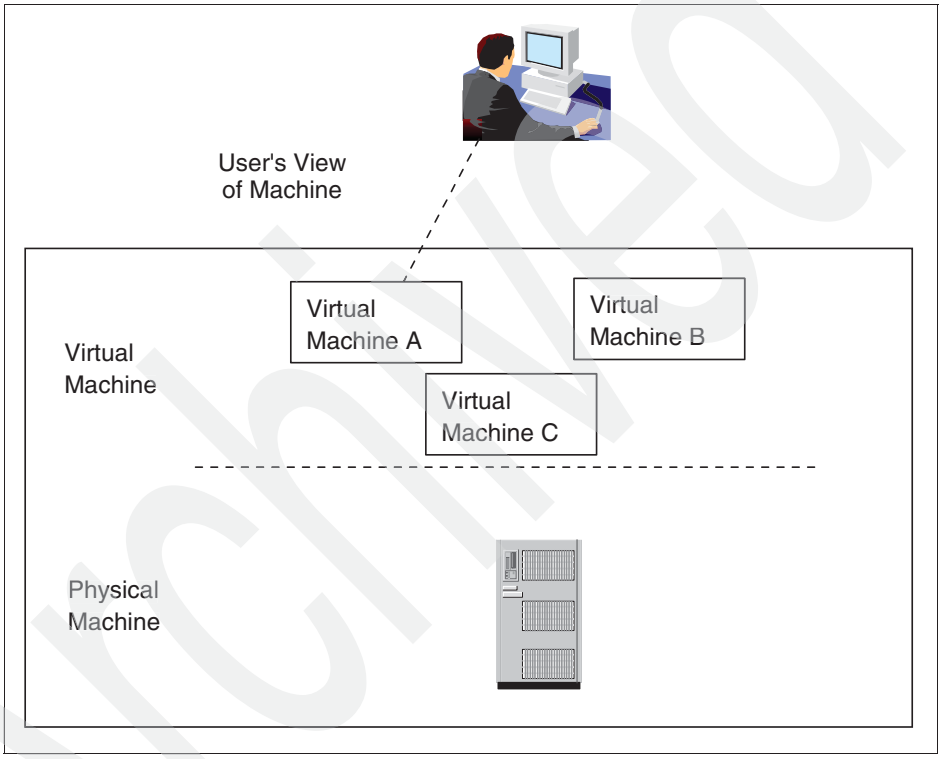


Figure 24-1 Virtualization

Storage virtualization

Storage virtualization is the combination of the capacity of multiple storage controllers into a single resource with a single view of the storage resources. This virtual layer between the physical storage devices and the users or host application provides the ability to conceal the physical infrastructure from the application and user. A benefit of storage virtualization is the ability to add, upgrade, or remove space and disks without the applications or the users service being affected.

Archived

Network virtualization

Network virtualization enables administrators to manage portions of a network that might be shared among different enterprises as virtual networks, while still continuing to preserve the isolation of traffic and resource utilization. Network virtualization also enables the administrator to prioritize traffic across the network to ensure the optimum performance for vital business applications and processes. This includes technologies such as virtual private networks (VPNs), HiperSockets™, virtual networks, and virtual LANs.

24.6 Virtual Router Redundancy Protocol (VRRP)

Virtual Router Redundancy Protocol (VRRP) was issued to the IETF by IBM, Ascend Communications, Microsoft, and Digital Equipment Corporation in April 1998 and is documented in RFC 3768. Its status is a proposed standard.

24.6.1 Introduction

The use of a statically configured default route is quite popular for host IP configurations. It minimizes configuration and processing inefficiencies on the end host and is supported by virtually every IP implementation. This mode of operation is likely where dynamic host configuration protocols (such as 3.7, “Dynamic Host Configuration Protocol (DHCP)” on page 130) are deployed, which typically provide configuration for an end host IP address and default

Archived

gateway. However, this creates a single point of failure. Loss of the default router results in a catastrophic event, isolating all end hosts that are unable to detect any alternate path that may be available. Figure 24-2 illustrates VRRP.

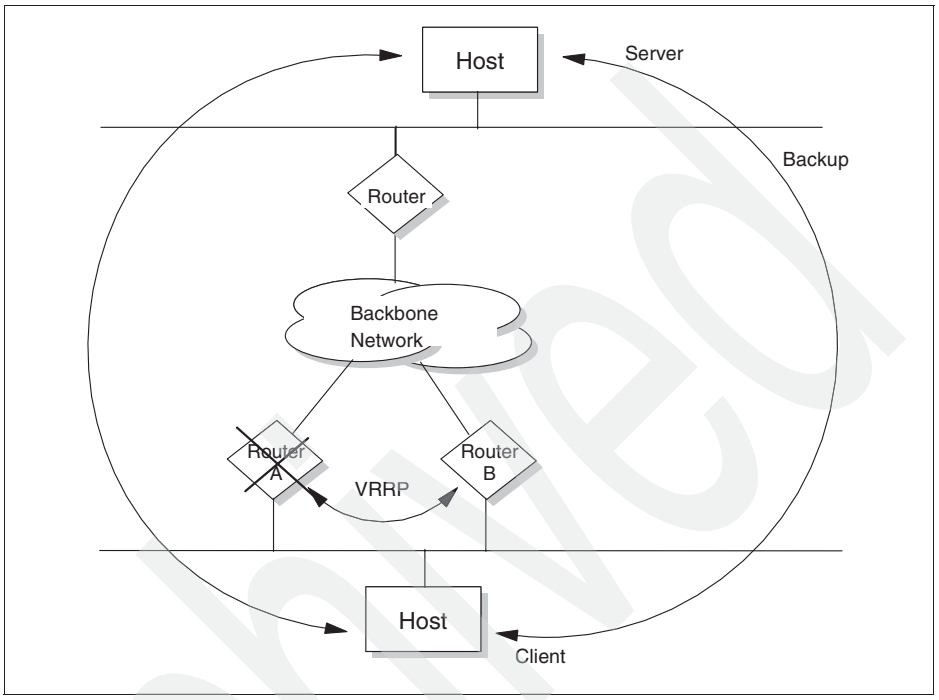


Figure 24-2 An illustration of VRRP

VRRP is designed to eliminate the single point of failure inherent in the static, default, routed environment. VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. The VRRP router controlling the IP addresses associated with a virtual router is called the master, and it forwards packets sent to these IP addresses. The election process provides dynamic fail-over in the forwarding responsibility if the master becomes unavailable. Any of the virtual router's IP addresses on a LAN can then be used as the default first hop router by end hosts. The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end host (see router discovery protocols in 3.2, “Internet Control Message Protocol (ICMP)” on page 109).

24.6.2 VRRP definitions

Some terms used in VRRP are:

VRRP router A router running the Virtual Router Redundancy Protocol. It can participate in one or more virtual routers.

Virtual router An abstract object managed by VRRP that acts as a default router for hosts on a shared LAN. It consists of a virtual router identifier and a set of associated IP addresses depending on the definition, across a common LAN. A VRRP router can back up one or more virtual routers.

IP address owner The VRRP router that has the virtual router's IP addresses as real interface addresses. This is the router that, when up, responds to packets addressed to one of these IP addresses for ICMP pings, TCP connections, and so on.

Primary IP address An IP address selected from the set of real interface addresses. One possible selection algorithm is to always select the first address. VRRP advertisements are always sent using the primary IP address as the source of the IP packet.

Virtual router master The VRRP router that is assuming the responsibility of forwarding packets sent to the IP addresses associated with the virtual router and answering ARP requests for these IP addresses. Note that if the IP address owner is available, it will always become the master.

Virtual router backup The set of VRRP routers available to assume forwarding responsibility for a virtual router if the current master fails.

24.6.3 VRRP overview

VRRP specifies an election protocol to provide the virtual router function described earlier. All protocol messaging is performed using IP multicast datagrams (see Chapter 6, “IP multicast” on page 237), thus the protocol can operate over a variety of multiaccess LAN technologies supporting an IP multicast. Each VRRP virtual router has a single well-known MAC address allocated to it. The virtual router MAC address is used as the source in all periodic VRRP messages sent by the master router to enable bridge learning in an extended LAN.

A virtual router is defined by its virtual router identifier (VRID) and a set of IP addresses. A VRRP router can associate a virtual router with its real addresses on an interface and can also be configured with additional virtual router mappings and priority for virtual routers it is willing to back up. The mapping between VRID and addresses must be coordinated among all VRRP routers on a LAN. However, there is no restriction against reusing a VRID with a different address mapping on different LANs.

The scope of each virtual router is restricted to a single LAN. To minimize network traffic, only the master for each virtual router sends periodic VRRP advertisement messages. A backup router will not attempt to preempt the master unless it has higher priority. This eliminates service disruption unless a more preferred path becomes available. It is also possible to administratively prohibit all preemption attempts. The only exception is that a VRRP router will always become master of any virtual router associated with addresses it owns. If the master becomes unavailable, the highest priority backup will transition to master after a short delay, providing a controlled transition of the virtual router responsibility with minimal service interruption.

The VRRP protocol design provides rapid transition from master to backup to minimize service interruption and incorporates optimizations that reduce protocol complexity while guaranteeing controlled master transition for typical operational scenarios. The optimizations result in an election protocol with minimal runtime state requirements, minimal active protocol states, and a single message type and sender. The typical operational scenarios are defined to be two redundant routers or distinct path preferences among each router, or both. A side effect when these assumptions are violated (for example, more than two redundant paths all with equal preference) is that duplicate packets can be forwarded for a brief period during the master election. However, the typical scenario assumptions are likely to cover the vast majority of deployments, loss of the master router is infrequent, and the expected duration in master election convergence is quite small (< 1 second). Therefore, the VRRP optimizations represent significant simplifications in the protocol design while incurring an insignificant probability of brief network degradation.

24.6.4 Sample configuration

Figure 24-3 shows a simple example network with two VRRP routers implementing one virtual router.

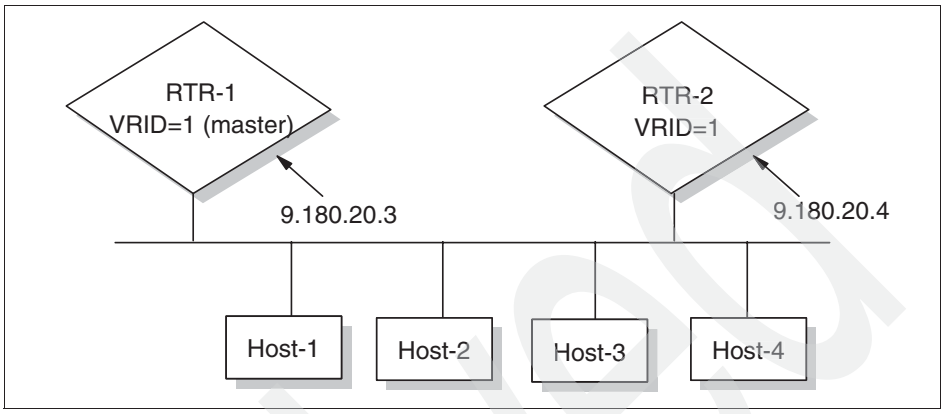


Figure 24-3 VRRP simple configuration example

This configuration shows a very simple VRRP scenario. In this configuration, the end hosts install a default route to the IP address of virtual router #1 (IP address 9.180.20.3) and both routers run VRRP. The router on the left becomes the master for virtual router #1 (VRID=1), and the router on the right is the backup for virtual router #1. If the router on the left fails, the other router takes over virtual router #1 and its IP addresses, and provides uninterrupted service for the hosts. Note that in this example, IP address 9.180.20.4 is not backed up by the router on the left. IP address 9.180.20.4 is only used by the router on the right as its interface address. In order to back up IP address 9.180.20.4, a second virtual router needs to be configured. This is shown in Figure 24-4.

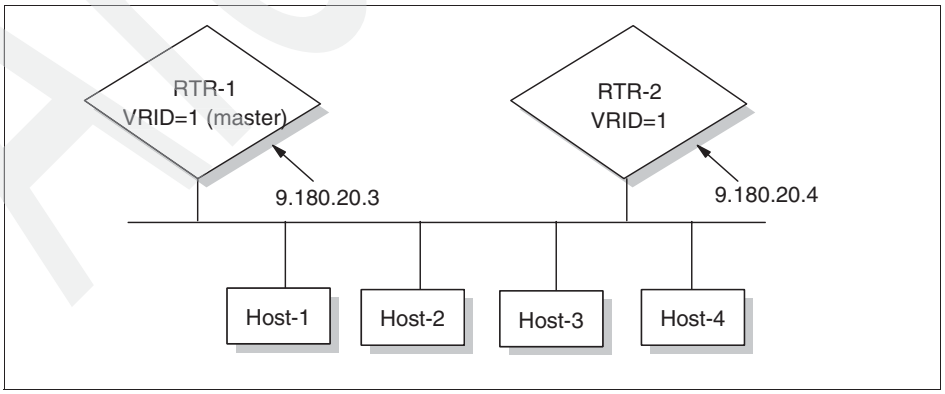


Figure 24-4 VRRP simple load-splitting configuration example

Figure 24-4 on page 918 shows a configuration with two virtual routers with the hosts splitting their traffic between them. This example is expected to be very common in actual practice. In this configuration, half of the hosts install a default route to virtual router #1 (IP address of 9.180.20.3), and the other half of the hosts install a default route to virtual router #2 (IP address o f 9.180.20.4). This has the effect of load balancing the traffic from the hosts through the routers, while also providing full redundancy.

24.6.5 VRRP packet format

The purpose of the VRRP packet is to communicate to all VRRP routers the priority and the state of the master router associated with the virtual router ID. VRRP packets are sent encapsulated in IP packets. They are sent to the IPv4 multicast address assigned to VRRP. The IP address, as assigned by the IANA for VRRP, is 224.0.0.18. This is a link local scope multicast address. Routers must not forward a datagram with this destination address regardless of its TTL (see 3.1, “Internet Protocol (IP)” on page 68). The TTL must be set to 255. A VRRP router receiving a packet with the TTL not equal to 255 must discard the packet. Figure 24-5 shows the VRRP packet format.

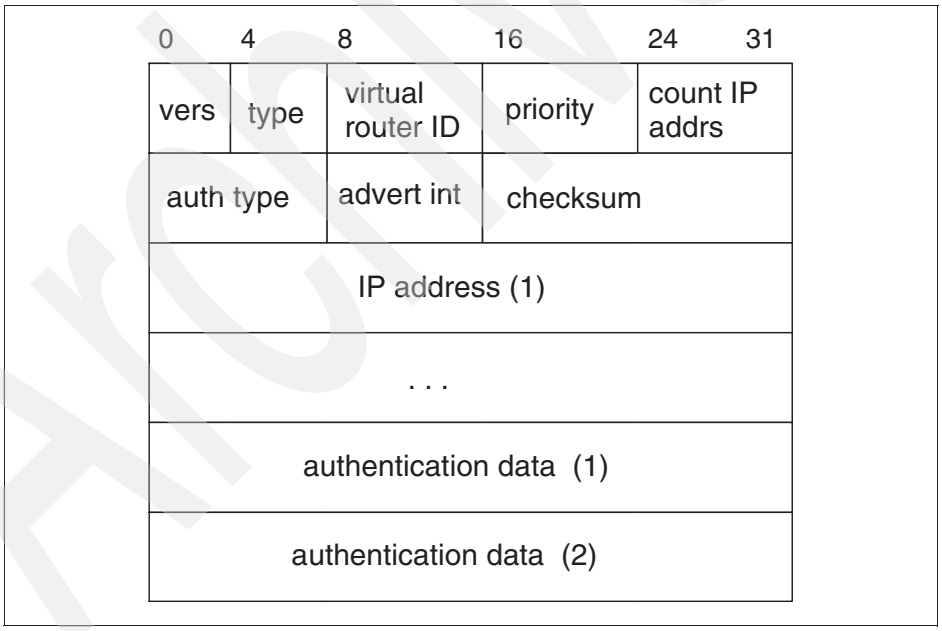


Figure 24-5 VRRP packet format

The fields of the VRRP header are defined as follows:

Version The version field specifies the VRRP protocol version of this packet. (In RFC 3768, the version is 2.)

Type The type field specifies the type of this VRRP packet. The only packet type defined in this version of the protocol is 1.

Virtual router ID (VRID) The virtual router identifier (VRID) field identifies the virtual router for which this packet is reporting the status.

Priority The priority field specifies the sending VRRP router's priority for the virtual router. Higher values equal higher priorities. The priority value for the VRRP router that owns the IP addresses associated with the virtual router must be 255. VRRP routers backing up a virtual router must use priority values between 1-254. The default priority value for VRRP routers backing up a virtual router is 100. The priority value zero (0) has special meaning, indicating that the current master has stopped participating in VRRP. This is used to trigger backup routers to quickly transition to master without having to wait for the current master to time out.

Count IP addrs The number of IP addresses contained in this VRRP advertisement.

Auth type The authentication type field identifies the authentication method being utilized. Authentication type is unique on a per interface basis. The authentication type field is an 8-bit unsigned integer. A packet with unknown authentication type or that does not match the locally configured authentication method must be discarded. The authentication methods currently defined are:

0 - No authentication

1 - Simple text password

2 - IP authentication header

Advertisement interval (Adver Int)
The default is 1 second. This field can be used for troubleshooting misconfigured routers.

Checksum This is used to detect data corruption in the VRRP message.

IP address(es)	One or more IP addresses that are associated with the virtual router.
Authentication data	The authentication string is currently only used for simple text authentication.

24.7 Round-robin DNS

Early solutions to address load balancing were often located at the point where host names are translated into actual IP addresses: the Domain Name System (see 12.1, “Domain Name System (DNS)” on page 426). By rotating through a table of alternate IP addresses for a specific service, some degree of load balancing is achieved. This approach is often called round-robin DNS. The advantages of this approach are that it is protocol-compliant and transparent both to the client and the destination host. Also, it is performed only once at the start of the transaction.

Unfortunately, this approach is sometimes defeated because intermediate name servers and client software (including some of the most popular browsers) cache the IP address returned by the DNS service and ignore an expressly specified time-to-live (TTL) value (see 3.1, “Internet Protocol (IP)” on page 68), particularly if the TTL is short or zero. As a result, the balancing function provided by the DNS is bypassed, because the client continues to use a cached IP address instead of resolving it again. Even if a client does not cache the IP address, basic round-robin DNS still has limitations:

- It does not provide the ability to differentiate by port.
- It has no awareness of the availability of the servers.
- It does not take into account the workload on the servers.

RFC 1794 discusses DNS support for load balancing and mentions round-robin DNS.

24.8 Alternative solutions to load balancing

There are many vendors currently offering load balancing hardware or software products. The techniques used vary widely and have advantages and disadvantages.

24.8.1 Network Address Translation

Network Address Translation (NAT) works by modifying the source and target IP addresses in the inbound client-to-server packets and restoring the IP address to the original values in the outbound server-to-client packets. (Refer to 3.1.7, “Network Address Translation (NAT)” on page 89.)

Note that if NAT is to be transparent to the server, eliminating the need for specialized agent code on the server, all packets sent back to the client must pass back through the load balancer in order to restore the IP addresses originally used by the client, as shown in Figure 24-6. This is a significant inefficiency, which will have a varying impact on the load balancer and the servers whose resources it manages.

This added processing charge and latency can mean network delay, and queuing delay in the load balancer itself. This drastically limits the potential scalability of NAT solutions. To overcome such delays, the capacity of the load balancer must not only be sufficient to handle both inbound and outbound packets, but also be able to cope with the disproportionately higher volume of outbound traffic.

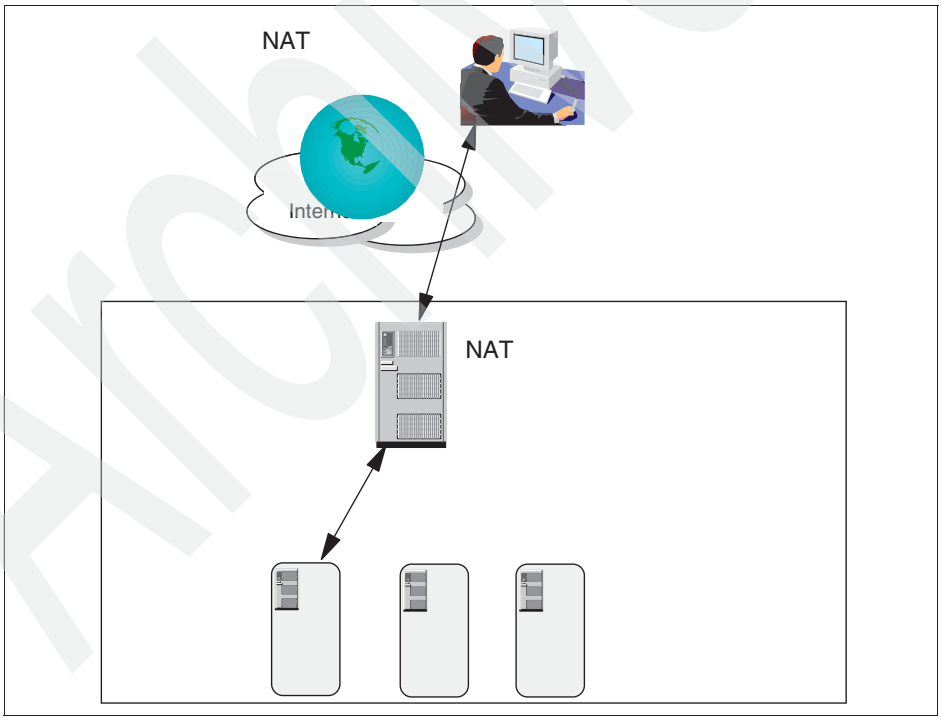


Figure 24-6 Network Address Translation

As shown in Figure 24-6 on page 922, NAT offerings sometimes enforce the need to see both inbound and outbound requests by obliging the NAT device to be installed as a bridge (without permitting bridges of any other kind), thus forcing the servers on to what is essentially a private segment. This can complicate installation, because it requires a significant physical change to existing networks. All traffic for those servers must pass through the load balancer whether the traffic is to be load balanced or not.

The one advantage of NAT as originally conceived (the ability to forward packets to remote destinations across a wide area network) cannot be usefully deployed because the wide area network connection is behind the bridge and, therefore, can only be within the site's private network. Additionally, the same NAT device must still be the only exit from the wide area network link.

To attempt to overcome these limitations, some NAT solutions add to the overall inefficiency that is fundamental to NAT by providing add-ons. For example, the capability to map one port address to another. Refer to “Network Address Port Translation (NAPT)” on page 93.

To check if a server is up, NAT-based load balancing solutions need to sacrifice an actual client request, and so a server outage is typically perceived only as a result of a timeout of one of these real client requests.

NAT devices often only map affinity or *stickiness* based on the client's IP address, and not at the port level. This means that after a client has contacted a server, all traffic from that client that is intended for other applications is forwarded to the same server. This drastically restricts configuration flexibility, in many cases rendering the sticky capability unusable in the real world.

24.8.2 Encapsulation

Another approach to load balancing is proxies that encapsulate packets rather than modifying them, and then pass them to the server. This approach has some merit, particularly because it permits the load balancer to forward traffic across a wide area network, unlike the bridging NAT solutions. But other implementations use encapsulation for all traffic, and this requires an agent of the load balancer to be installed on each server. This agent reverses the encapsulation process. As a result, the choice of server platform is, by definition, restricted to the platforms for which the server agent is available. Also, like NAT, it entails further processing of the packet, which increases the likelihood that it will not be scalable to the levels required for major sites.

Encapsulation is discussed in further detail in RFC 1701, which appears to be updated by RFC 2784.

24.9 RFCs relevant to this chapter

The following RFCs provide detailed information about availability, scalability, and load balancing as presented throughout this chapter:

- RFC 1794 – DNS Support for Load Balancing (April 1995)
- RFC 3768 – Virtual Router Redundancy Protocol (VRRP) (April 2004)
- RFC 2784 – Generic Routing Encapsulation (GRE) (March 2000)
- RFC 1701 – Generic Routing Encapsulation (GRE) (October 1994)

Archived

24

Multiprotocol Label Switching

This chapter provides an overview of the Multiprotocol Label Switching (MPLS) process and Generalized MPLS (GMPLS) architecture.

This chapter describes the following topics:

- The ideas behind processing MPLS
- An explanation of the differences between conventional, connectionless routing and flow routing
- A definition of the terminology for MPLS concepts
- A summary of the benefits of MPLS flow routing
- A review of the details of MPLS protocols
- An introduction to GMPLS

© Copyright IBM Corp. 1989-2006. All rights reserved.

A.1 MPLS: An introduction

The idea behind MPLS was to emulate some property of circuit-switching network over a packet network, and to strike a happy middleground between extreme connection-oriented switching and pure connectionless routing service.

The theory for the idea of “mixing types to meet in the middle” was based on the observation that a sequence of correlated packets exist for stream and multimedia applications. We wanted to process them in the same routing path in a uniform fashion in order to guarantee quality of service (QoS). And we did not want to repeatedly examine and process those packet headers. The idea is feasible because we note that the headers in those related packets are the same or similar because those related packets in a stream desire consistent and similar processing treatment.

Multiprotocol Label Switching (MPLS) follows the same idea and comes up with new techniques to make a pseudo (and short-term) connection in a path (or subpath) for a sequence of correlated IP packets. The technology was proposed in RFC 3031. The Multiprotocol Label Switching (MPLS) standard represents the effort in the continued evolution of multilayer switching.

Generalized MPLS or GMPLS extends MPLS to encompass time-division (for example, SONET/SDH), wavelength (lambdas), and spatial switching (for example, incoming port or fiber to outgoing port or fiber). The focus of GMPLS is on the control plane of these various layers to dynamically provision resources and to provide network survivability using protection and restoration techniques.

A.1.1 Conventional routing versus MPLS forwarding mode

First, let us describe how the new paradigm shift might help in improving QoS performance by comparing conventional routing to the MPLS forwarding mode.

In an MPLS environment, conventional layer-3 or network-layer routing (that is, IP routing) is used to determine a path through the network. After the path is determined, data packets are then switched through each node as they traverse the network.

Conventional routing mode

In a traditional, connectionless network, every router runs a layer-3 routing algorithm. As a packet traverses through the network, each router along the path makes an independent forwarding decision for that packet. Using information contained in the packet header, as well as information obtained from the routing algorithm, the router chooses a “next hop” destination for the packet. In an IP network, this process involves matching the destination address stored in the IP header of each packet with the most specific route obtained from the IP routing

table. This comparison process determines the next hop destination for the packet. This analysis and classification of the layer-3 header can be processor-intensive. In a traditional connectionless environment, this activity occurs at every node along the end-to-end path.

MPLS forwarding model

In an MPLS environment, optimum paths through the network are identified in advance. Then, as data packets enter the MPLS network, ingress devices use information in the layer-3 header to assign the packets to one of the predetermined paths. This assignment is used to append a *label* referencing the end-to-end path into the packet. The label accompanies the data packet as it traverses the network. Subsequent routers along the path use the information in the label to determine the next hop device. Because these devices only manipulate information in the label, processor-intensive analysis and classification of the layer-3 header occurs only at the ingress point.

A.1.2 Benefits

In additional to reducing the processing requirements on devices in the core of the network, MPLS has a number of additional advantages over conventional layer-3 routing, which we describe in the following sections.

Traffic engineering

Traffic engineering is the process of selecting network paths so that the resulting traffic patterns achieve a balanced utilization of resources.

Routing based on conventional Interior Gateway Protocol (IGP) algorithms might select network paths that result in unbalanced resource utilization. In these environments, some network resources are overused, while others are underused. A limited degree of engineering can be provided by manipulating the IGP metrics associated with network links. However, this effort is difficult to manage in environments with a large number of redundant paths.

To achieve the benefits of traffic engineering, MPLS can be used in-conjunction with IGP algorithms. MPLS provides the ability to specify the specific route data packets should use to traverse the network. This explicit routing of data packets ensures that a particular stream of data uses a specific path. By monitoring and managing these data streams, efficient utilization of network resources can be achieved. Explicit routing has been available through the source routing options of traditional IP routing. However, because this is a processor-intensive activity, its usage has been limited. MPLS makes the efficient use of explicit routing possible.

MPLS also provides the ability to analyze fields outside the IP packet header when determining the explicit route for a data packet. For example, the network administrator can develop traffic flow policies based on how or where a packet entered the network. In a traditional network, this information is only available at the ingress point. The additional analysis provides the administrator with a higher level of control, resulting in a more predictable level of service.

Quality of service routing

QoS routing is the ability to choose a route for a particular data stream so that the path provides a desired level of service. These levels of service can specify acceptable levels of bandwidth, delay, or packet loss in the network. This provides the intelligence to deliver different levels of service based on overall network policies.

Providing a network path delivering a desired QoS often requires the use of explicit routing. For example, it is straightforward to allocate a path for a particular stream requiring a specific bandwidth allocation. However, it is possible that the combined bandwidth of multiple streams may exceed existing capacity. In this scenario, individual streams, even those between the same ingress and egress nodes, might need to be individually routed. This requires a finer level of granularity than that provided by standard traffic engineering.

There are two approaches to providing QoS routing in an MPLS environment:

- The MPLS label contains class of service (CoS) information. As traffic flows through the network, this information can be used to intelligently prioritize traffic at each network hop.
- The MPLS network can provision multiple paths between ingress and egress devices. Each path is engineered to provide a different level of service. Traffic is then intelligently assigned to an appropriate path as it enters the network.

These approaches simply classify packets into a class of service category. Local network administration policies determine the service provided to each category.

Multiprotocol support

The Multiprotocol Label Switching standard provides support for existing network-layer protocols, including IPv4, IPv6, IPX, and AppleTalk. The standard also provides link layer support for Ethernet, token ring, FDDI, ATM, frame relay, and point-to-point links. Activities continue to extend this standard to other protocols and network types.

MPLS is not limited to a specific link layer technology; it can function on any media over which network layer packets can pass.

A.1.3 Terminology

The following sections define the terms that are used with MPLS.

Forwarding equivalency class (FEC)

An FEC is a group of layer-3 packets that are forwarded in the same manner. All packets in this group follow the same network path and have the same prioritization. Packets within an FEC can have different layer-3 header information. However, to simply make a forwarding decision, these packets are indistinguishable.

Common examples of FEC groups are:

- A set of packets that have the same most specific route in the IP routing table.
- A set of packets that have the same most specific route in the IP routing table and the same IP type of service setting.

In an MPLS network, an FEC is identified by a label.

Label and labeled packet

As stated previously, a label identifies a unique FEC. MPLS devices forward all identically labeled packets in the same way.

A label is locally significant between a pair of MPLS devices. It represents an agreement between the two devices describing the mapping between a label and an FEC. The fact that labels are locally significant enhances the scalability of MPLS into large environments, because the same label need not be used at every hop.

The MPLS label can be located at different positions in the data frame, depending on the layer-2 technology used for transport. If the layer-2 technology supports a label field, the MPLS label is encapsulated in the native label field. In an ATM network, the VPI/VCI fields can be used to store an MPLS label. Similarly, the DLCI field can be used to store an MPLS label in frame relay networks.

If the layer-2 technology does not natively support a label, the MPLS label resides in an encapsulation header appended specifically for this purpose. The header is located between the layer-2 header and the IP header. This use of a dedicated header permits MPLS service over any layer-2 technology (see Figure A-1 on page 930, which depicts the 32-bit MPLS header).

Archived

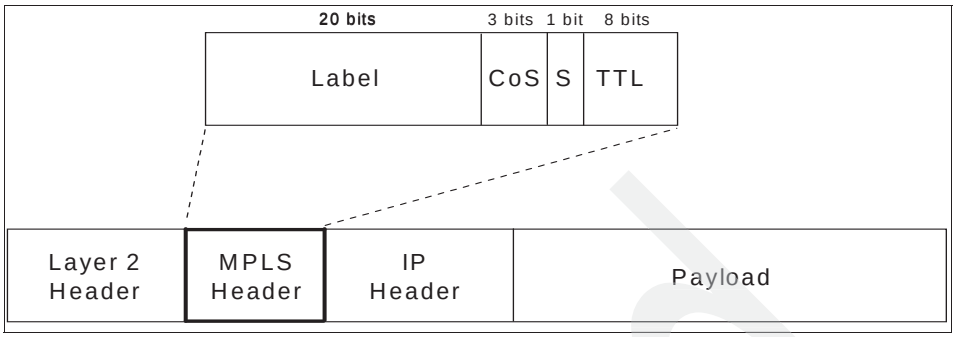


Figure A-1 The 32-bit MPLS header

The contents of the MPLS header include:

- A label field that contains the value of the MPLS label.
- A CoS field that can be used to affect the queuing and discard algorithms applied to the packet as it traverses the network.
- A S (stack) field that supports a hierarchical label stack.
- A TTL (time-to-live) field that supports conventional IP TTL functionality.

A labeled packet is a packet into which a label has been encoded. To support enhanced MPLS functions, the packet might contain more than one label. This is known as a label stack. The stack establishes an ordered relationship between individual labels. The stack is implemented using the last-in, first-out model. This feature is further discussed in A.2.3, “Label stack and label hierarchies” on page 934.

Label stack router (LSR)

A label stack router is an MPLS node that is also capable of forwarding native layer-3 packets. There are two important types of LSRs in an MPLS network:

- An *ingress node* connects the MPLS network with a node that does not execute MPLS functionality. The ingress node handles traffic as it enters the MPLS network.
- An *egress node* connects the MPLS network with a node that does not execute MPLS functionality. The egress node handles traffic as it leaves the MPLS network.

Archive

Next hop label forwarding entry (NHLFE)

An NHLFE is used by an MPLS node to forward packets. There is at least one NHLFE for each FEC flowing through the node. Each node is responsible for maintaining an NHLFE information base containing the following information:

- The packet's next hop address
- The operation performed on the label stack:
  - Replace the label at the top of the stack with a specified new label. This is known as *popping* the old label and *pushing* a new label.
  - Pop the label at the top of the stack.
  - Replace the label at the top of the stack with a specified new label, and then push one or more specified new labels onto the label stack. When this action is complete, the stack will contain at least two MPLS labels.
- The data link encapsulation used to transmit the packet (optional)
- The label stack encoding used to transmit the packet (optional)
- Any other information needed in order to properly process the packet

Incoming label map (ILM)

The ILM is used by an MPLS node to forward labeled packets. The label in an incoming packet is used as a reference to the ILM. The ILM information allows the node to select a set of NHLFEs containing forwarding instructions.

The ILM can map a label to a group of NHLFEs. This provides the ability to load balance over multiple equal-cost paths.

FEC-to-NHLFE map (FTN)

The FTN is used by an MPLS node to process packets that arrive unlabeled, but need to be labeled before forwarding. An unlabeled data packet is assigned a specific FEC at the ingress MPLS node. This FEC is used as a reference to the FTN. The FTN map allows the node to select a set of NHLFEs containing forwarding instructions. This activity is performed at the ingress node of the MPLS network.

The FTN can map a label to a group of NHLFEs. This provides the ability to load balance over multiple equal cost paths.

Label swapping

Label swapping is the process used by an MPLS node to forward a data packet to the next hop device. This process is used regardless of whether the packet arrives labeled or unlabeled. The process is similar to the method used in ATM and frame relay networks to forward traffic through a virtual circuit.

Label switched path (LSP)

An LSP represents a set of MPLS nodes traversed by packets belonging to a specific FEC. The set is an ordered, unidirectional list. Traffic flows from the node at the head-end of the list toward the node at the tail-end of the list.

Label stack and label hierarchies

A labeled packet can contain more than one label. The labels are maintained in a last-in, first-out stack. The stack implements an ordered hierarchy among the set of labels.

This hierarchy is used when an MPLS node delivers a packet to a partner MPLS node, but the nodes are not consecutive routers on the hop-by-hop path for the packet. In this situation, a tunnel is created between the two MPLS nodes. The tunnel is implemented as an LSP and label switching is used to forward traffic through the tunnel.

A.2 MPLS network processing

The primary goal of MPLS is the integration of label swapping paradigms with traditional network layer routing. This integration bring efficiencies in data forwarding as well as positioning the network for advanced QoS functions.

A.2.1 Label swapping

Label swapping is the process used by an MPLS node to forward a data packet to the next hop device. This process is used regardless of whether the packet arrives labeled or unlabeled. The process is similar to the method used in ATM and frame relay networks to forward traffic through a virtual circuit.

Forwarding a labeled packet

An MPLS node examines the label at the top of the stack of an incoming packet. It uses the ILM to map the label to an NHLFE. The NHLFE indicates where to forward the packet and the operation to perform on the label stack. Using this information, the node encodes a new label stack and forwards the resulting packet.

Forwarding an unlabeled packet

An MPLS node examines the network layer header and any other pertinent information required to determine an FEC. The node uses the FTN to map the FEC to an NHLFE. Processing is now identical to a labeled packet. The NHLFE indicates where to forward the packet and the operation to perform on the label

Archived

stack. Using this information, the node encodes a new label stack and forwards the resulting packet.

Figure A-2 depicts label swapping in an MPLS environment.

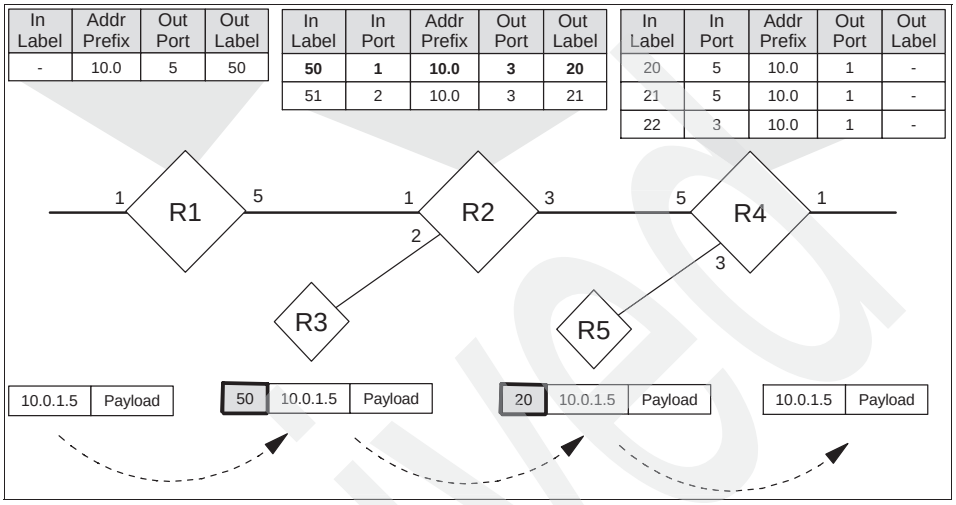


Figure A-2 Label swapping in an MPLS environment

Note: In a label swapping environment, the next hop router is always determined from MPLS information. This might cause the packet to traverse a different path than the one obtained using conventional routing algorithms.

Penultimate hop popping

This is the ability to pop an MPLS label at the penultimate node rather than at the egress node. From an architectural perspective, this type of processing is permitted. The purpose of a label is to forward a packet through the network to the egress node. After the penultimate node has decided to send the packet to the egress node, the label no longer has any function. It does not need to be included in the packet.

The penultimate node pops the stack and forwards the packet based on the next hop address obtained from the NHLFE. When the egress node receives the packet, one of two activities occur:

- The packet contains a label. This occurs when the penultimate node processed a packet with at least two labels. In this scenario, the label now at the top of the stack is the label the egress node needs to process to make a forwarding decision.

- The packet does not contain a label. In this scenario, the LSP egress receives a standard network layer packet. The node uses the local IP routing table to make a forwarding decision.

A.2.2 Label switched path (LSP)

An LSP represents a set of MPLS nodes traversed by packets belonging to a specific FEC. The set is an ordered, unidirectional list. Traffic flows from the node at the head-end of the list toward the node at the tail-end of the list. The LSP for the traffic flow shown in Figure A-2 on page 933 is <R1, R2, R4>.

In an MPLS network, LSPs can be established in one of two ways:

- Independent LSP control: Each LSR makes an independent decision to bind a label to an FEC. It then distributes the label to its peer nodes. This is similar to conventional IP routing; each node makes an independent decision as to how to forward a packet.
- Ordered LSP control: An LSR binds a label to a particular FEC only if it is the egress LSR for that FEC, or if it has already received a label binding for that FEC from its next hop for that FEC. In an environment implementing traffic engineering policies, ordered LSP control is used to ensure that traffic in a particular FEC follows a specific path.

Section A.2.5, “Label distribution protocols” on page 938 describes the procedures used to exchange label information in an MPLS environment.

A.2.3 Label stack and label hierarchies

A label stack (FILO) is used in tunneling between the two MPLS nodes. The tunnel is implemented as an LSP and label switching is used to forward traffic through the tunnel.

The set of traffic sent through the tunnel constitutes an FEC. Each LSR in the tunnel must assign a label to that FEC.

To send a packet through the tunnel, the tunnel ingress node pushes a label understood by the tunnel egress node onto the label stack. The tunnel ingress node then pushes a label understood by the next hop node and forwards the data packet through the tunnel.

Archive

For example, a network might contain an LSP <R1, R2, R3, R4>. In this example, R2 and R3 are not directly connected, but are peers endpoints of an LSP tunnel. The actual sequence of LSRs traversed through the network is <R1, R2, R21, R22, R3, R4>. Figure A-3 shows this configuration.

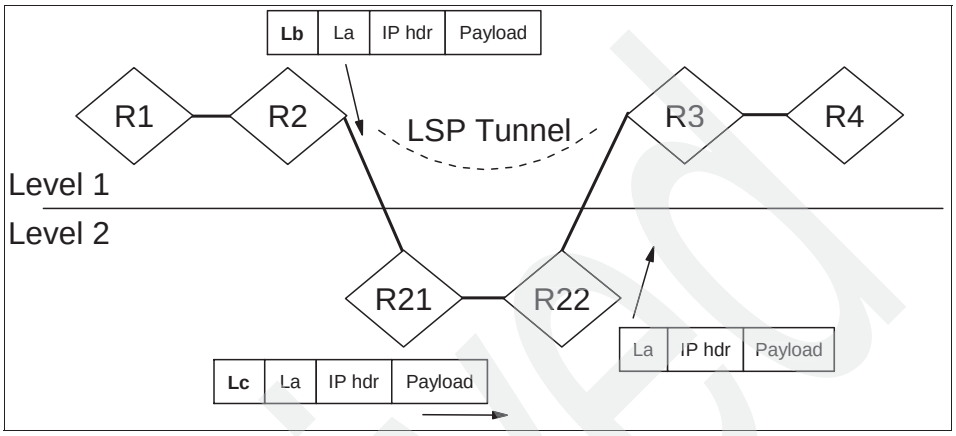


Figure A-3 LSP tunnels

A packet traversing this network travels along a level-1 LSP: <R1, R2, R3, R4>. Then, when traveling from R2 to R3, uses a level-2 LSP: <R2, R21, R22, R3>. From the perspective of the level-1 LSP, R2's peer devices are R1 and R3. From the level-2 perspective, R2's peer device is R21.

Using this diagram, the following actions occur when a packet is sent through the LSP tunnel:

1. R2 receives a labeled packet from R1. The packet contains a single label. The depth of the label stack is one.
2. R2 pops this label and pushes a label understood by R3. This label is called La.
3. R2 must also include a label understood by R21. R2 pushes the label on top of the existing level-1 label. This label is called Lb. The label stack contains two entries.
4. R2 forwards the packet to R21.
5. R21 pops the level-2 label (Lb) appended by R2 and pushes a level-2 label understood by R22. This label is called Lc. R21 does not process the level-1 label. The label stack contains two entries.
6. R21 forwards the packet to R22.

7. R22 reviews the level-2 label appended by R21 and realizes it is the penultimate hop in the R2-R3 tunnel. R22 pops the level-2 label (Lc) and forwards the packet to R3. The label stack contains one entry.

A.2.4 MPLS stacks in a BGP environment

The network shown in Figure A-4 shows three autonomous systems. The environment contains two classes of IP routing:

- Each autonomous system runs an IGP to maintain connectivity within the AS. For example, R2, R21, R22, and R3 use OSPF to maintain routes within AS 2.
- Each autonomous systems runs BGP to maintain connectivity between autonomous systems. For example, border routers R1, R2, R3, and R4 use BGP to exchange inter-AS routing information.

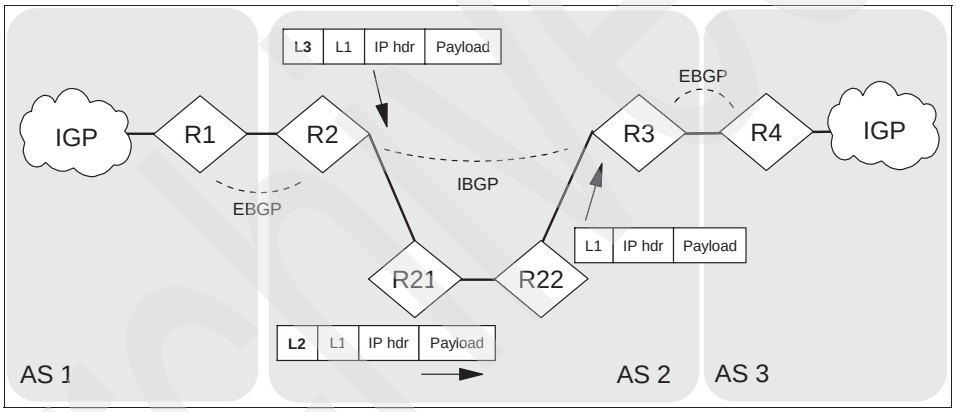


Figure A-4 Connecting autonomous systems in an MPLS environment

In this sample network, it is desirable to avoid distributing BGP-learned routes to devices which are not BGP border routers (for example, R21, R22). This minimizes the CPU processing required to maintain the IP routing table on these devices. It also eliminates the need to run a BGP routing algorithm on these devices.

An MPLS LSP stack can be used to implement this environment. In this configuration, BGP routes are distributed only to BGP peers, and not to interior routers that lie along the hop-by-hop path between peers. LSP tunnels are configured so that:

- Each peer distributes a label for each address prefix that it distributes through BGP. These labels are distributed to peers within the same AS.

- The IGP maintains a host route for each BGP border router. Each interior router distributes a label for the host route to each IGP neighbor.

Consider a situation where R2 receives an unlabeled packet destined for a network connected through AS 3. The packet might have originated from a LAN segment locally connected to R2 or another LAN segment within AS 2. The packet would have been previously labeled if it had originated in AS1.

1. R2 searches the local IP forwarding table to determine the most specific route for the required destination address. The route will have be learned through BGP. The BGP next hop will be R3.
2. R3 has previously bound a label for the longest match and distributed this label to R2. This label is called L1.
3. Because all devices within AS 2 participate in the IGP, a route to R3 appears in the routing table of all devices within AS 2:
   - R22 has previously bound a label for R3 and distributed this label to R21. This label is called L2.
   - R21 has previously bound a label for R3 and distributed this label to R2. This label is called L3.
4. R2 prepares the data packet destined for AS 3 by creating a label stack. The initial entry on the stack is created by pushing the L1 label. The top entry on the stack is created by pushing the L3 label. The labeled packet is then sent to the next hop, R21.
5. R21 receives the labeled packet and reviews the top entry in the stack. Using the information in the NHLFE, R21 replaces the L3 label in the stack with the L2 label. The labeled packet is then sent to the next hop, R22.
6. R22 receives the labeled packet and reviews the top entry. Because R22 is the penultimate hop on the R2-R3 tunnel, R22 pops the L2 label on the stack and forwards the data packet to R3. The label stack now contains a single entry as it is forwarded to R3.
7. R3 receives the labeled data packet and reviews the L1 label on the stack. Using information in the NHLFE, R3 replaces the old label with a label bound by R4 and forwards the packet.

Note: Whenever an MPLS node pushes a label on to an already labeled packet, the new label must correspond to an FEC whose LSP egress is the node that assigned the new label.

A.2.5 Label distribution protocols

A label distribution protocol is a set of procedures that allows one MPLS node to distribute labels to other peer nodes. This specification is used by an LSR to notify another LSR of an assigned label and its associated meaning. This exchange establishes a common agreement between peers.

Each MPLS node participates in a local IGP to determine the network topology and populate the routing table. Label distribution protocols use this information to establish labels. After a distribution protocol has run in each node, the entire MPLS network has a complete set of paths and associated labels.

Label distribution protocols also encompass any negotiations between peers needed to learn the MPLS capabilities of each peer.

Types of label distribution protocols

The MPLS architecture does not specify a required distribution protocol nor does it assume there is only a single protocol. Because of this, there are a number of different standards under development. These standards can be placed into one of two categories.

Extensions to existing protocols

Proposals have been made to existing protocols so that label distribution information is included within existing data flows. Two examples of this are:

- BGP extensions: In many cases, FECs are used to identify address prefixes distributed by BGP peers. It might be advantageous to have these same devices distribute MPLS labels. Further, the use of BGP route reflectors to distribute labels can provide significant scalability enhancements.
- RSVP extensions: This proposal enhances the RSVP standard to include support for establishing and distributing LSP information. This enables the allocation of resources along the end-to-end path.

Development of new protocols

New protocols are also being developed with the sole purpose of distributing labels. These stand-alone protocols do not rely on the presence of specific routing protocols at every hop along the path. This is useful in situations in which an LSP must traverse nodes that do not support one of the existing protocols that has been extended to include label distribution functions.

Label distribution methods

There are two methods to initiate communication between MPLS nodes to exchange label information:

- Downstream-on-demand: An LSR can request a label binding for a particular FEC. The request is made to the next hop MPLS node for that FEC.
- Unsolicited downstream: An LSR can distribute bindings to LSRs that have not explicitly requested the information.

  Both of these distribution techniques can be used in the same network at the same time. For a given set of peers, the upstream LSR and the downstream LSR must agree on the technique to be used.

A.2.6 Stream merge

Stream merge is the aggregation of a large number of data flows into a single downstream flow. The device performing the merge consolidates the individual streams so that they are treated as a single stream by subsequent MPLS nodes. The merged stream is represented by a single label. After the merged packets are transmitted, any information that the packets arrived with different incoming labels is lost.

Stream merge is a major component of MPLS scalability.

A.3 Emulating Ethernet over MPLS networks

As MPLS gaining popularity, there is a trend to carry non-IP traffic over MPLS network directly. One example is the encapsulation method for transport of Ethernet over MPLS networks.

Instead of using Ethernet cable or switch, we use MPLS as the underlying transport. The Ethernet service operates on top of the MPLS network layer as illustrated in Figure A-5 on page 940.

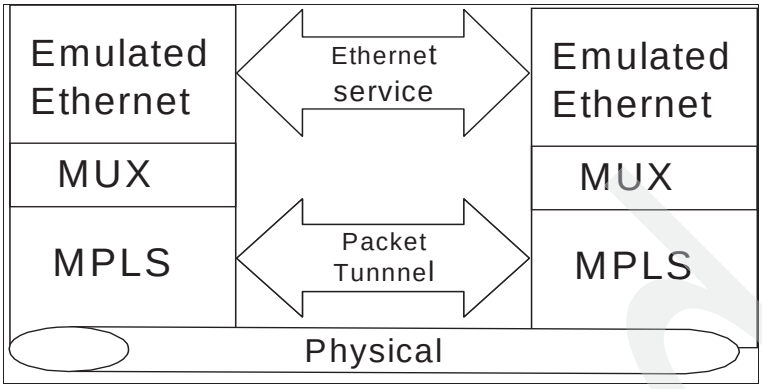


Figure A-5 Encapsulation of Ethernet over MPLS

When emulating Ethernet over MPLS networks:

- The Label Distribution Protocol (LDP) sets up a pseudo wire (as opposed to a real Ethernet cable wire, or a Ethernet switch) on MPLS.
- The Ethernet pseudo wire carries the Ethernet/802.3 protocol data units (PDUs) over MPLS.
- The pseudo wire emulation consists of the destination address, source address, length/type, MAC client data, and padding extracted from a MAC frame as a concatenated octet sequence in their original order.
- Usually, only point-to-point Ethernet connections are emulated as the broadcast operation in a real Ethernet.
- The emulation can provide multiplexing and demultiplexing at the consumer edge routers so that aggregrated traffic is between provider edge MPLS routers.
- An Ethernet PW operates in one of two modes, raw mode or tagged mode:
  - In tagged mode, each frame *must* contain at least one 802.1Q Virtual LAN tag, and the tag value is meaningful to the NSPs at the two pseudo wire termination points.
  - On a raw mode PW, a frame *may* contain an 802.1Q VLAN tag, but if it does, the tag is not meaningful to the NSPs, and passes transparently through them.
- In addition to providing a bridge of Ethernet frames over MPLS, the pseudo wire methods can provide frame delivery in sequence as well as other QoS controls.

We expect that over time other protocols such as ATM will be emulated as well.

A.4 Generalized Multiprotocol Label Switching (GMPLS)

GMPLS extends MPLS into new transmission network types (for example, to include optical networks).

The optical network is becoming an ideal IP transport infrastructure in core and metro networks due to its potentially unlimited bandwidth. It is desirable to move toward a two-layer architecture that transports IP traffic directly over the optical network because:

- Increased data network complexities are driving architectures that provide cost-effective and scalable solutions. These solutions must also provide performance optimization features.
- Challenges include rapid and effective bandwidth provisioning and protection/restoration. Some of these requirements are addressed by the MPLS standard. These same business requirements are applicable to the underlying optical transport network (OTN).

As a result, standards bodies are starting to develop specifications for incorporating MPLS functions into layer-1 optical cross-connect (OXC) devices as well as layer-2 transmissions. Standards efforts are addressing the linking of IP services directly to the optical networks that carry data, allowing these networks to take advantage of the routing intelligence now embedded in IP headers.

Note: The focus of GMPLS efforts are mainly in the control plane where new signaling techniques are described in the GMPLS architecture document (GMPLS architecture is documented in RFC 3945).

GMPLS provides a common control plane for devices that operate in multiple domains, such as time-division-multiplexing, packet, or wavelength switching. The common control plane simplifies operation management through automatic provision of connections, network resources, and quality-of-service setups.

A.4.1 Benefits

There are several benefits to extending MPLS functions into optical networks:

- The activity can leverage techniques developed for MPLS to provide a methodology for real-time provisioning of optical channels. This promotes faster development of these technologies.
- The outcome can provide a uniform approach to network management for both data and optical environments. This simplifies overall management efforts.

- The effort can provide a single architecture, allowing an LSP to traverse a mix of router and OXC devices. This positions the network to provide real bandwidth on demand facilities.
- The effort positions IP routers for the eventual incorporation of high-capacity dense wave division multiplexing (DWDM) capabilities.
- It provides new services such as Optical Virtual Private Network or Optical VPN, and Differentiated Circuit with service-level assurance.
- It enhances bandwidth brokering with instant bandwidth on demand capability.
- It also standardizes ways to handle connections through automatic provisioning.
- GMPLS extends the MPLS protocol so that it can control not only routers, but also:
  - Dense wavelength division multiplexing (DWDM) systems
  - Add drop multiplexers
  - Optical cross-connects
  - Optical switches

A.4.2 MPLS and GMPLS comparison in OTN environment

The extension of MPLS functions into optical network is based on a number of similarities between the two types of networks:

- In MPLS, an LSP describes the point-to-point path traversed by a set of labeled packets. In GMPLS, an optical channel trail is used to describe the point-to-point optical connection between two access points.
- In an MPLS network, an LSR establishes a relationship between an <input port, input label> tuple and <output port, output label> tuple. Similarly, in an optical transport network, an OXC establishes a relationship between an <input port, input optical channel> tuple and <output port, output optical channel> tuple. When established, these relationships cannot be altered by the contents of any data packet.
- In an MPLS network, an LSR discovers, distributes, and maintains state information associated with the network. An OXC is responsible for these same functions in an OTN.
- In an MPLS network, an LSR is responsible for creating and maintaining LSPs associated with existing traffic engineering policies. An OXC is responsible for the same functions in an OTN.
- In an MPLS network, an LSP is unidirectional. In an OTN, an optical channel trail is also unidirectional.

Archived

In a GMPLS environment, it is possible to represent the fiber within the OTN as a set of links, each link representing a set of channels. An IP routing protocol (with extensions) distributes information about the optical network topology, available bandwidth, and other pertinent state information. The information computes explicit routes for optical channel trails. MPLS distribution protocols then creates these trails.

As development of this technology continues, important distinctions between an MPLS network and an OTN must be addressed:

- In an MPLS network, forwarding information is carried as part of the label contained in each data packet. In an OTN, switching information is implied from the wavelength or optical channel.
- In an OTN, there is no concept of label merging. An OXC cannot merge several wavelengths into one wavelength.
- An OXC cannot perform label push or pop operations. In an optical domain, the analog of a label is a wavelength. There is no concept of pushing or popping wavelengths in current optical technologies.

A.4.3 How does GMPLS work?

GMPLS extends MPLS to layer-1 optical transmissions. It combines IP and optical control planes such that routers, packet switches, and optical switches are aware of each other.

GMPLS deals with several interfaces:

- Packet Switch Capable (PSC) interface, which is the only interface supported by MPLS
- Layer-2 Switch Capable (L2SC) interface, which requires additional functions to recognize frames and cells
- Time Division Multiplexing Capable (TDM) interface to forward data based on the data's time slot
- Lambda Switch Capable (LSC) interface, which work on individual wavelengths as in optical cross-connects
- Fiber Switch Capable (FSC) interface, which work on individual or multiple fibers

GMPLS uses a generalized label object to accommodate a widening scope of MPLS into the optical and time-division domain. A generalized label contains enough information to allow the receiving node to set up the path.

A generalized label only carries a single level of label; that is, it is non-hierarchical. When multiple levels of label (LSPs within LSPs) are required, each LSP must be established separately.

Note: GMPLS uses signaling to reserve and set up routing or connection paths. The process is call signaling, which means that a separate control link is used to convey requests and parameters.

A special control link carries the route control information. The separation of control and data paths gives the meaning of “outband” signaling.

GMPLS extends routing protocols to disseminate non-packet switch characteristics. For example, it uses a link descriptor for encoding and transmission rate, available and reservable bandwidth, protection- and restoration-related characteristics, and so on.

GMPLS is a family of protocols including:

- Link Management Protocol for neighbor discovery
- Signaling protocol for route selection and path setup

A.4.4 Link Management Protocol (LMP)

LMP provides the following functions:

- Neighbor identification and link verification
- Shared link group
- Fault isolation
- Topology-aware networks

Note: LMP allows GMPLS networks to obtain peer-to-peer awareness such that every participating Label Switch Router (LSR) sees the entire network design topology.

The two core procedures of LMP are control channel management and link property correlation. In addition, fault management and control channel activation are necessary.

Control channel management

Control channel management is used to establish and maintain control channels between adjacent nodes. This is done using a “Config” message exchange and a

fast keepalive mechanism between the nodes. The latter is required if lower-level mechanisms are not available to detect control channel failures.

If the LMP fast keepalive is used over a control channel, LMP Hello messages *must* be exchanged over the control channel. Other LMP messages *may* be transmitted over any of the active control channels between a pair of adjacent nodes. One or more active control channels can be grouped into a logical control channel for signaling, routing, and link property correlation purposes.

Link property correlation

Link property correlation is used to synchronize the Traffic Engineering (TE) link properties and verify the TE link configuration. The link property correlation function of LMP is designed to aggregate multiple data links (ports or component links) into a TE link and to synchronize the properties of the TE link.

As part of the link property correlation function, a “LinkSummary” message exchange is defined. The LinkSummary message includes:

- The local and remote Link_Ids
- A list of all data links that make up the TE link
- Various link properties

A “LinkSummaryAck” or “LinkSummaryNack” message *must* be sent in response to the receipt of a LinkSummary message indicating agreement or disagreement on the link properties.

Additional LMP procedures are defined for link connectivity verification and fault management. These procedures are particularly useful when the control channels are physically diverse from the data links.

Link connectivity verification is used for data plane discovery, Interface_Id exchange (Interface_Ids are used in GMPLS signaling, either as port labels or component link identifiers, depending on the configuration), and physical connectivity verification. This is done by sending Test messages over the data links and TestStatus messages back over the control channel.

Note: The Test message is the only LMP message that must be transmitted over the data link. The ChannelStatus message exchange is used between adjacent nodes for both the suppression of downstream alarms and the localization of faults for protection and restoration.

Fault management

The LMP fault management procedure is based on a ChannelStatus message exchange that uses the following messages:

- The ChannelStatus message is sent unsolicited and is used to notify an LMP neighbor about the status of one or more data channels of a TE link.
- The ChannelStatusAck message is used to acknowledge receipt of the ChannelStatus message.
- The ChannelStatusRequest message is used to query an LMP neighbor for the status of one or more data channels of a TE link.
- The ChannelStatusResponse message is used to acknowledge receipt of the ChannelStatusRequest message and indicate the states of the queried data links.

Control channel

A control channel transports the reservation and routing information for traffic path setup:

- To establish a control channel, the destination IP address on the far end of the control channel must be known. This knowledge can be manually configured or automatically discovered.
- Control channels exist independently of TE links and multiple control channels can be active simultaneously between a pair of nodes. Individual control channels can be realized in different ways; one might be implemented in-fiber while another one might be implemented out-of-fiber.
- There are four LMP messages used to manage individual control channels:
  - Config (Configuration)
  - ConfigAck (Positive acknowledgement)
  - ConfigNack (Negative Acknowledgement)
  - Hello messages
- Control channel activation begins with a parameter negotiation exchange using Config, ConfigAck, and ConfigNack messages. The contents of these messages are built using LMP objects.
- LMP uses a “Hello” protocol to send “Hello packets” exchanges over each control channel to maintain LMP connectivity. The LMP Hello protocol is intended to be a lightweight keepalive mechanism that will react to control channel failures rapidly so that IGP Hellos are not lost and the associated link-state adjacencies are not removed unnecessarily.

- Before sending Hello messages, the HelloInterval and HelloDeadInterval parameters *must* be agreed upon by the local and remote nodes. These parameters are exchanged in the Config message.
  - The HelloInterval indicates how frequently LMP Hello messages will be sent, and is measured in milliseconds (ms). For example, if the value were 150, the transmitting node sends the Hello message at least every 150 ms.
  - The HelloDeadInterval indicates how long a device waits to receive a Hello message before declaring a control channel dead, and is measured in milliseconds (ms).
  - The HelloDeadInterval *must* be greater than the HelloInterval, and *should* be at least three times the value of HelloInterval.
  - If the fast keepalive mechanism of LMP is not used, the HelloInterval and HelloDeadInterval parameters *must* be set to zero.
- When a node has either sent or received a ConfigAck message, it can begin sending Hello messages. After it has sent a Hello message and received a valid Hello message, it moves to the up state.
- If, however, a node receives a ConfigNack message instead of a ConfigAck message, the node *must* not send Hello messages and the control channel must *not* move to the up state.

For detailed descriptions of the state transitions and messaging flows, refer to the RFC 4204 LMP specifications.

A.4.5 Signaling for route selection and path setup

GMPLS adds signaling capabilities. When the routed is calculated, it uses resource ReSerVation Protocol Traffic Engineering (RSVPTE) or Constraint-based Routing Label Distribution Protocol (CR-LDP) to set up the LSP as it would with MPLS.

A generalized label request also indicates the type of switching that is being requested on a link. This field normally is consistent across all links of an LSP.

Possible actions include:

- Signal into the network with a request, specifying the type of connections desired.
- Change a path to use new capacity.
- Set up direct path when appropriate.
- Build in restoration.

Note: GMPLS allows SP to dynamically provision extra resources and to provide redundancy. They are for the purpose of protection and restoration.

A generalized label request can contain a number of parameters:

- LSP encoding type (for example, SONET, SDH)
- Payload type (client layer)
- Requested local protection on each link (1+1, 1:N)
- Requested concatenation and transparency (only for SONET/SDH)

As a further example, the generalized label request carries an LSP encoding parameter, called LSP Encoding Type. The LSP Encoding Type represents the nature of the LSP, and not the nature of the links that the LSP traverses. Sample values include those shown in Table A-1.

Table A-1 Sample values

Value	Type
1	Packet
2	Ethernet
3	ANSI/ETSI PDH
4	Reserved
5	SDH ITU-T G.707 / SONET ANSI T1.105
6	Reserved
7	Digital Wrapper
8	Lambda (photonic)
9	Fiber
10	Reserved
11	Fibre Channel

Consider an LSP signaled with “lambda” encoding. It is expected that such an LSP would be supported with no electrical conversion and no knowledge of the modulation and speed by the transit nodes. Other formats normally require framing knowledge, and field parameters are broken into the framing type and speed.

A.4.6 GMPLS considerations

Because GMPLS is attempting to have a generic framework (one size fits all) to cover all possible cases, it actually introduces unexpected complexity and new issues. We provide a brief introduction to certain issues common in GMPLS.

Performance consideration

With bidirectional LSPs, both the downstream and upstream data paths are established using a single set of signaling messages. This reduces the setup latency to essentially one initiator-terminator round-trip time plus processing time, and limits the control processing cost to the same number of messages as a unidirectional LSP.

Contention resolution

Contention for labels can occur between two bidirectional LSP setup requests traveling in opposite directions. This contention occurs when both sides allocate the same resources (labels) at effectively the same time. There is also an additional potential race for conditions in assignment of resources, which decreases the overall probability of successfully establishing the bidirectional connection.

If there is no restriction on the labels that can be used for bidirectional LSPs and if there are alternate resources, both nodes will pass different labels upstream and there is no contention. However, if there is a restriction on the labels that can be used for the bidirectional LSPs (for example, if they must be physically coupled on a single I/O card), or if there are no more resources available, the contention must be resolved by other means.

When a request message is processed, the signaling process resolves any contentions for the same label when two bidirectional LSP setup requests are traveling in opposite directions:

- The node with the higher node ID wins the contention and it must issue a PathErr/NOTIFICATION message with a “Routing problem/Label allocation failure” indication.
- Upon receipt of such an error, the node should try to allocate a different Upstream label (and a different Suggested Label if used) to the bidirectional path.
- If no other resources are available, the node must proceed with standard error handling.

To reduce the probability of contention, you can impose a policy that the node with the lower ID never suggests a label in the downstream direction and always accepts a Suggested Label from an upstream node with a higher ID.

Furthermore, because the labels can be exchanged using LMP, an alternative local policy can further be imposed such that (with respect to the higher numbered node's label set) the higher numbered node can allocate labels from the high end of the label range while the lower numbered node allocates labels from the low end of the label range.

This mechanism would augment any close packing algorithms that might be used for bandwidth (or wavelength) optimization. One special case that should be noted when using RSVP and supporting this approach is that the neighbor's node ID might not be known when sending an initial Path message. When this case occurs, a node should suggest a label chosen at random from the available label space.

Addressing and routing

GMPLS reuses IP routing and addressing models. A routing domain is made up of GMPLS enabled nodes.

For scalability purposes, GMPLS implements two enhancements to addressing the links:

- Unnumbered links: Logic links that carry TE properties but do not need an address.
- Link bundling: A single logical link to represent a number of links (either physical or logical).

Path computation allows proprietary algorithms for distributed and centralized computation. It is essentially a constraint-based routing problem. The routes are selected according to quality of service (QoS) requirements and the resource capability along the path.

Routing protocols are usually extended from OSPF-TE and IS-IS-TE. They must allow the auto-discovery of network topology, advertise resource availability, and accommodate unnumbered links or bundled links.

A.4.7 GMPLS examples

The Optical Internetworking Forum (OIF) is defining a User Network Interface (UNI) for the GMPLS specification. It is called the OIF UNI.

- OIF UNI allows neighbor discovery and channel assignment at the Synchronous Digital Hierarchy or STS level.
- The control interface can be an out-of-band Ethernet interface or an in-band command carried in the SDH/Sonet processing charge.

- The UNI specifies a way for a client, called a User Network Interface Client (UNI-C), to invoke transport network services with a User Network Interface Network (UNI-N), a device on another provider's network.
- It allows the following actions:
  - Connection creation
  - Connection deletion
  - Connection status inquiry

Implementation model

In a particular implementation, generic functions in the GMPLS control plane are mapped into different functional modules:

- Resource management module (RMM)

  RMM is used for routing and wavelength assignment (RWA), topology, and resource discovery quality of service (QoS) support.
- Connection module (CM)

  CM is used for lightpath signaling and maintenance. In the GMPLS control plane, CR-LDP or RSVP-TE is used for signaling.
- Protection/restoration module (PRM)

  PRM provides fault monitoring and fast protection/restoration.
- Main module (MM)

  The MM objective is to receive the incoming messages and work closely with other modules to process the requests.

Required TE building blocks

GMPLS can have multiple building blocks for Traffic Engineering. However, only the following building blocks are mandatory:

- A generic label request format
- Labels for TDM, LSC, and FSC interfaces, known as generalized labels
- Specific parameters for each transmission technology

Archived

A.5 RFCs relevant to this chapter

The following RFCs provide detailed information about MPLS and GMPLS as presented throughout this chapter.

Basic MPLS specifications are documented in RFC 3031, while basic GMPLS architecture is documented in RFC 3945.

- RFC 3031 – Multiprotocol Label Switching Architecture (January 2001)
- RFC 3036 – LDP Specification (January 2001)
- RFC 3212 – Constraint-Based LSP Setup using LDP (January 2002)
- RFC 4003 – GMPLS Signaling Procedure for Egress Control (February 2005)
- RFC 4201 – Link Building in MPLS Traffic Engineering (TE) (February 2005)
- RFC 4448 – Encapsulation Methods for Transport of Ethernet over MPLS Networks (April 2006)
- RFC4447 – Pseudowire Setup and Maintenance using LDP (April 2006)
- RFC 4328 – GMPLS Extensions for G.709 Optical Transport Networks Control (January 2006)
- RFC 3945 – Generalized Multi-Protocol Label Switching (GMPLS) Architecture (October 2004)
- RFC 4204 – Link Management Protocol (LMP) (October 2005)
- RFC 4208 – Generalized Multiprotocol Label Switching (GMPLS) User-Network Interface (UNI): Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model (October 2005)
- RFC 4257 – Framework for Generalized Multi-Protocol Label Switching (GMPLS)-based Control of Synchronous Digital Hierarchy/Synchronous Optical Networking (SDH/SONET) Networks (December 2005)
- RFC 4258 – Requirements for Generalized Multi-Protocol Label Switching (GMPLS) Routing for the Automatically Switched Optical Network (ASON) (November 2005)
- RFC 4397 – A Lexicography for the Interpretation of Generalized Multiprotocol Label Switching (GMPLS) Terminology within the Context of the ITU-T's Automatically Switched Optical Network (ASON) (February 2006)

Archived

Abbreviations and acronyms

AAA	Authentication, Authorization, and Accounting
AAL	ATM Adaptation Layer
AFS	Andrew File System
AH	Authentication Header
AIX	Advanced Interactive Executive Operating System
API	Application Programming Interface
APPN	Advanced Peer-to-Peer Networking®
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BIND	Berkeley Internet Name Domain
BNF	Backus-Naur Form
BRI	Basic Rate Interface
BSD	Berkeley Software Distribution
CA	Certification Authority
CBC	Cipher Block Chaining
CCITT	Comité Consultatif International Télégraphique et Téléphonique (now ITU-T)
CDMF	Commercial Data Masking Facility
CERN	Conseil Européen pour la Recherche Nucléaire
CGI	Common Gateway Interface
CHAP	Challenge Handshake Authentication Protocol
CICS®	Customer Information Control System
CIDR	Classless Inter-Domain Routing
CIX	Commercial Internet Exchange
CLNP	Connectionless Network Protocol
CORBA	Common Object Request Broker Architecture
COS	Class of Service
CPCS	Common Part Convergence Sublayer
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DARPA	Defense Advanced Research Projects Agency
DCE	Distributed Computing Environment
DCE	Data Circuit-terminating Equipment
DDN	Defense Data Network
DDNS	Dynamic Domain Name System
DEN	Directory-Enabled Networking
DES	Digital Encryption Standard
DFS	Distributed File Service
DHCP	Dynamic Host Configuration Protocol
DLC	Data Link Control

© Copyright IBM Corp. 1989-2006. All rights reserved.

DLCI Data Link Connection Identifier

DLL Dynamic Link Library

DLSw Data Link Switching

DLUR Dependent LU Requester

DLUS Dependent LU Server

DME Distributed Management Environment

DMI Desktop Management Interface

DMTF Desktop Management Task Force

DMZ Demilitarized Zone

DNS Domain Name System

DoD U.S. Department of Defense

DOI Domain of Interpretation

DOS Disk Operating System

DSA Digital Signature Algorithm

DSAP Destination Service Access Point

DSS Digital Signature Standard

DTE Data Terminal Equipment

DTP Data Transfer Process

DVMRP Distance Vector Multicast Routing Protocol

EBCDIC Extended Binary Communication Data Interchange Code

EGP Exterior Gateway Protocol

ESCON® Enterprise Systems Connection

ESP Encapsulating Security Payload

FDDI Fiber Distributed Data Interface

FQDN Fully Qualified Domain Name

FR Frame Relay

FTP File Transfer Protocol

GGP Gateway-to-Gateway Protocol

GMT Greenwich Mean Time

GSM Group Special Mobile

GUI Graphical User Interface

HDLC High-level Data Link Control

HMAC Hashed Message Authentication Code

HPR High Performance Routing

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IAB Internet Activities Board

IAC Interpret As Command

IANA Internet Assigned Numbers Authority

IBM International Business Machines Corporation

ICMP Internet Control Message Protocol

ICSS Internet Connection Secure Server

ICV Integrity Check Value

IDEA International Data Encryption Algorithm

IDLC Integrated Data Link Control

IDRP Inter-Domain Routing Protocol

IEEE Institute of Electrical and Electronics Engineers

IESG Internet Engineering Steering Group

IETF Internet Engineering Task Force

IGMP Internet Group Management Protocol

IGN IBM Global Network®

IGP Interior Gateway Protocol

IIOP Internet Inter-ORB Protocol

IKE Internet Key Exchange

IMAP	Internet Message Access Protocol
IMS	Information Management System
IP	Internet Protocol
IPC	Interprocess Communication
IPSec	IP Security Architecture
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IPX	Internetwork Packet Exchange
IRFT	Internet Research Task Force
ISAKMP	Internet Security Association and Key Management Protocol
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITSO	International Technical Support Organization
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector (was CCITT)
IV	Initialization Vector
JDBC™	Java Database Connectivity
JDK™	Java Development Toolkit
JES	Job Entry System
JIT	Java Just-in-Time Compiler
JMAPI	Java Management API
JVM	Java Virtual Machine
JPEG	Joint Photographic Experts Group
LAC	L2TP Access Concentrator
LAN	Local Area Network
LAPB	Link Access Protocol Balanced
LCP	Link Control Protocol
LDAP	Lightweight Directory Access Protocol
LE	LAN Emulation (ATM)
LLC	Logical Link Layer
LNS	L2TP Network Server
LPD	Line Printer Daemon
LPR	Line Printer Requester
LSAP	Link Service Access Point
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunneling Protocol
MAC	Message Authentication Code
MAC	Medium Access Control
MD2	RSA Message Digest 2 Algorithm
MD5	RSA Message Digest 5 Algorithm
MIB	Management Information Base
MILNET	Military Network
MIME	Multipurpose Internet Mail Extensions
MLD	Multicast Listener Discovery
MOSPF	Multicast Open Shortest Path First
MPC	Multi-Path Channel
MPEG	Moving Pictures Experts Group
MPLS	Multiprotocol Label Switching
MPOA	Multiprotocol over ATM
MPTN	Multiprotocol Transport Network
MS-CHAP	Microsoft Challenge Handshake Authentication Protocol
MTA	Message Transfer Agent

MTU Maximum Transmission Unit

MVS Multiple Virtual Storage Operating System

NAT Network Address Translation

NBDD NetBIOS Datagram Distributor

NBNS NetBIOS Name Server

NCF Network Computing Framework

NCP Network Control Protocol

NCSA National Computer Security Association

NDIS Network Driver Interface Specification

NetBIOS Network Basic Input/Output System

NFS Network File System

NIC Network Information Center

NIS Network Information Systems

NIST National Institute of Standards and Technology

NMS Network Management Station

NNTP Network News Transfer Protocol

NRZ Non-Return-to-Zero

NRZI Non-Return-to-Zero Inverted

NSA National Security Agency

NSAP Network Service Access Point

NSF National Science Foundation

NTP Network Time Protocol

NVT Network Virtual Terminal

ODBC Open Database Connectivity

ODI Open Datalink Interface

OEM Original Equipment Manufacturer

ONC Open Network Computing

ORB Object Request Broker

OSA Open Systems Adapter

OSI Open Systems Interconnect

OSF Open Software Foundation

OSPF Open Shortest Path First

OS/2 Operating System/2®

OS/400 Operating System for the AS/400 Platform

PAD Packet Assembler/ Disassembler

PAP Password Authentication Protocol

PDU Protocol Data Unit

PGP Pretty Good Privacy

PI Protocol Interpreter

PIM Protocol Independent Multicast

PKCS Public Key Cryptosystem

PKI Public Key Infrastructure

PNNI Private Network-to-Network Interface

POP Post Office Protocol

POP Point-of-Presence

PPP Point-to-Point Protocol

PPTP Point-to-Point Tunneling Protocol

PRI Primary Rate Interface

PSDN Packet Switching Data Network

PSTN Public Switched Telephone Network

PVC Permanent Virtual Circuit

QLLC Qualified Logical Link Control

QoS Quality of Service

RACF® Resource Access Control Facility

RADIUS Remote Authentication Dial-In User Service

RAM Random Access Memory

Archived

RARP Reverse Address Resolution Protocol

RAS Remote Access Service

RC2 RSA Rivest Cipher 2 Algorithm

RC4 RSA Rivest Cipher 4 Algorithm

REXEC Remote Execution Command Protocol

RFC Request for Comments

RIP Routing Information Protocol

RIPE Réseaux IP Européens

RISC Reduced Instruction-Set Computer

ROM Read-only Memory

RPC Remote Procedure Call

RSH Remote Shell

RSVP Resource Reservation Protocol

RTCP Realtime Control Protocol

RTP Realtime Protocol

SA Security Association

SAP Service Access Point

SDH Synchronous Digital Hierarchy

SDLC Synchronous Data Link Control

SET Secure Electronic Transaction

SGML Standard Generalized Markup Language

SHA Secure Hash Algorithm

S-HTTP Secure Hypertext Transfer Protocol

SLA Service Level Agreement

SLIP Serial Line Internet Protocol

SMI Structure of Management Information

S-MIME Secure Multipurpose Internet Mail Extension

SMTP Simple Mail Transfer Protocol

SNA System Network Architecture

SNAP Subnetwork Access Protocol

SNG Secured Network Gateway (former product name of the IBM eNetwork Firewall)

SNMP Simple Network Management Protocol

SOA Start of Authority

SONET Synchronous Optical Network

SPI Security Parameter Index

SSL Secure Sockets Layer

SSAP Source Service Access Point

SSP Switch-to-Switch Protocol

SSRC Synchronization Source

SVC Switched Virtual Circuit

TACACS Terminal Access Controller Access Control System

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TFTP Trivial File Transfer Protocol

TLPB Transport-Layer Protocol Boundary

TLS Transport Layer Security

TOS Type of Service

TRD Transit Routing Domain

TTL Time to Live

UDP User Datagram Protocol

UID Unique Identifier

URI Uniform Resource Identifier

URL Uniform Resource Locator

UT Universal Time

VC Virtual Circuit

VM Virtual Machine Operating System

VPN Virtual Private Network

VRML	Virtual Reality Modeling Language
VRRP	Virtual Router Redundancy Protocol
VTAM®	Virtual Telecommunications Access Method
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WSP	Wireless Session Protocol
WTP	Wireless Transaction Protocol
WAN	Wide Area Network
WWW	World Wide Web
XDR	External Data Representation
XML	Extensible Markup Language
X11	X Window System Version 11
X.25	CCITT Packet Switching Standard
X.400	CCITT and ISO Message-handling Service Standard
X.500	ITU and ISO Directory Service Standard
X.509	ITU and ISO Digital Certificate Standard
3DES	Triple Digital Encryption Standard

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 961. Note that some of the documents referenced here may be available in softcopy only.

- *DCE Cell Design Considerations*, SG24-4746
- *Local Area Network Concepts and Products: LAN Architect*, SG24-4753
- *Virtualization and the On Demand Business*, REDP-9115

Other publications

These publications are also relevant as further information sources:

- CCITT Recommendations I.465 and V.120, “Data Terminal Equipment Communications over the Telephone Network with Provision for Statistical Multiplexing”, *CCITT Blue Book*, Volume VIII, Fascicle VIII.1, 1988
- International Telecommunication Union, “ISDN Data Link Layer Specification for Frame Mode Bearer Services,” ITU-T Recommendation Q.922, 1992
- “Information technology -- Protocol identification in the network layer,” ISO/IEC TR 9577, 1999
- Dr. Wei Liu, Next Generation Network (NGN) Lecture Series, December 2005

Online resources

These Web sites are also relevant as further information sources:

- TCP/IP and System z

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`
- TCP/IP and System p

  `http://www.ibm.com/systems/p/library/index.html`

© Copyright IBM Corp. 1989-2006. All rights reserved.

- TCP/IP and System i

  http://www.ibm.com/servers/eserver/iseries/tcpip/index.html
- TCP/IP and System x

  http://www.ibm.com/servers/eserver/support/xseries/allproducts/installing.html
- IBM developerWorks “Virtualization in a nutshell: A pattern point of view” article

  http://www.ibm.com/developerworks/grid/library/gr-virt/
- U.S. Department of Commerce Commercial Encryption Export Controls

  http://www.bis.doc.gov/Encryption/Default.htm
- Exchange Point

  http://www.ep.net
- Internet2

  http://www.internet2.edu
- IETF Web site

  http://www.ietf.org
- Internet Software Consortium

  http://www.isc.org
- 6NET project

  http://www.6net.org
- MOONv6 project

  http://www.moonv6.org/
- Global IPv6 Summit 2006

  http://www.ipv6.net.cn/2006
- 802.11 family of standards

  http://www.ieee802.org/11/
- Open Mobile Alliance (OMA) specifications

  http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
- Internet Corporation for Assigned Names and Numbers (ICANN)

  http://www.icann.org
- DEN initiative

  http://www.dmtf.org/standards/wbem/den/
  http://www.dmtf.org/standards/cim/

Archived

- Information about the most current release of DCE (Version 1.2.2) at the Open Group Web site

  http://www.opengroup.org/dce
- Information about AFS at the OpenAFS Web site

  http://www.openafs.org/
- Sendmail, Inc.

  http://www.sendmail.org
- Object Management Group

  http://www.omg.org
- WAP specification

  http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
- ITU - WG 11 (MPEG), MPEG-4 Overview, July 2000

  http://www.itu.int/ITU-D/tech/digital-broadcasting/kiev/References/mpeg-4.html
- Computer Science Lecture series, Volume 868/1994 (with a collections of IBM/Siemens Multimedia Workshop presentations)

  http://www.springer.com/west/home/computer/lncs?SGWID=4-164-22-1435374-0

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

Index

A

A5 algorithm 780
AAA security model 872
accounting 873
address allocation 131
Address Resolution (ATMARP and InATMARP) 47
Address Resolution Protocol (ARP)
  algorithm 122
  BOOTP 126
  cache 119
  Ethernet 31, 120
  Frame Relay 44
  IEEE 802.2 standard 31
  IEEE 802.x standards 120
  IPv6 354
  packet 120, 125
  Proxy-ARP 123
  reply 119
  request 119
algorithm
  block 779
  Diffie-Hellman 784
  digital signature 790, 837
  key-exchange 783
  public-key 781–783
  RSA 783
  stream 779
  symmetric 779–780
anycast 86
application-level gateway 796
arithmetic, modular 783
ARPANET 14
  Network Working Group 13
Assigned Numbers RFC 814
Asynchronous Transfer Mode 47
ATM 47, 51, 53, 56, 58
atm 50
ATM LAN emulation 56
ATMARP 47
ATM-Attached Host Functional Group (AHFG) 62
attacks 772
Authentication 773
authentication 36, 777, 779, 781, 786–787, 800, 848, 854, 872, 881
  IP Control Protocol (IPCP) 37
  IPCP 37
Authentication Header (AH)
  Authentication Data 815
  Authentication Data field 788
  combinations with ESP 823
  Flags field 813
  Fragment Offset 813
  header checksum 813
  header format 814
  HMAC-MD5-96 815
  HMAC-SHA-1-96 815
  Integrity Check Value (ICV) 815
  IP fragment 813
  IPv6 environment 817
  Keyed MD5 815
  mutable fields 813, 816
  Next Header field 814
  Payload Length 814
  replay protection 815
  Reserved field 814
  Security Parameter Index (SPI) 814
  Sequence Number 815
  Time To Live (TTL) 813
  transform 810
  transport mode 815, 824
  tunnel mode 815, 824
  Type of Service (TOS) 813
authentication method 834
authorization 773, 872
autoconfiguration 363

B

Berkeley Software Distribution (BSD) 13
block algorithm 779
Bluetooth 27
BOOTP forwarding 129
BOOTP server 125
Bootstrap Protocol (BOOTP) 126
  ARP 126
  BOOTPREQUEST message 129
  BOOTREPLY message 128

© Copyright IBM Corp. 1989-2006. All rights reserved.

DHCP interoperability 140
forwarding 84, 129
message format 127
TFTP 126
Border Gateway Protocol (BGP)
Version 4 (BGP-4) 98
bridge 11
broadcast
description 84
reverse path forwarding algorithm 85
brute-force attack 772, 780
bulk encryption 783

C

cache
ARP cache 119
ICMPv6 neighbor cache 356
CBC 779
CCITT 13
Cell 51
certificate 773, 832, 859
certificate authority 837, 881
certification authority (CA) 791
checksum
IPv6 333
chicken and egg problem 127
cipher 778
restricted 778
Cipher Block Chaining (CBC) 779
ciphertext 782
circuit-level gateway 796
Classical IP over ATM 50, 61
classless inter-domain routing (CIDR)
IP prefix 97
cleartext 777
client/server model 10
CLNP 40, 42
codebook 779
collision-resistant 785
combined tunnel 826
Commercial Internet Exchange (CIX) 17
Connections
ATM 47
FDDI 33
Frame Relay 41
ISDN 38
MPC+ 46
MPOA 60
PPP 35
SLIP 34
SONET 45
X.25 39
cryptanalysis 773, 777
cryptanalyst 793
cryptographic algorithm 778
cryptography 777, 785
strong 777
cryptography, strong 777
cryptosystem 783, 792

D

DARPA 13
Data Encryption Standard (DES) 779, 865
Data Link Connection Identifier (DLCI) 41
data-link layer 8
DCA 14
decryption 777–778
Default Forwarder Function Group (DFFG) 63
default router 63
Defense Advanced Research Projects Agency (DARPA) 13
Defense Communication Agency (DCA) 14
Demilitarized Zone (DMZ) 808
denial-of-service attack 772–773, 813, 830, 879
Department of Defense (DoD) 14
DES 779, 783, 792, 865
destination options extension header 817, 822
DHCPv6 371
dictionary attack 772
Diffie-Hellman 783–784, 793, 835–836, 839–840, 842, 844–845
digital certificate 791
digital envelope 787
digital signature 781, 787, 838–839
Digital Signature Standard 788, 831
discrete logarithms 783, 785
diskless host 126
Distributed Computing Environment (DCE) 469
DOI 833–834
Domain Name System (DNS) 24
IPv6 381
IPv6 extensions 368
IPv6 inverse lookups 368
resource record (RR)
format 368
resource record for IPv6 368

Archived

Domain of Interpretation (DOI) 833–834
DSA 790
DSS 788, 790
Dynamic Host Configuration Protocol (DHCP) 126, 130
address allocation 131
BOOTP forwarding 129
BOOTP interoperability 140
BOUND state 137
client/server interaction 134, 137–138
DHCPACK message 134
DHCPDECLINE message 134
DHCPDISCOVER message 134
DHCPINFORM message 134
DHCPNACK message 134
DHCPRELEASE message 134
DHCPREQUEST message 134
DHCPv6
DHCP Advertise 372
DHCP Reconfigure 372
DHCP Release 372
DHCP Reply 372
DHCP Request 372
DHCP Solicit 371
ICMPv6 364
INIT state 135
lease time 137
message format 132
message types 134
REBINDING state 137
RENEWING state 137
timer 137

E

e-business 862
ECB 779
Edge Device Functional Group (EDFG) 62
Electronic Codebook Mode (ECB) 779
Encapsulating Security Payload (ESP)
authentication 818, 824
Authentication Data 820
CDMF 819
combinations with AH 823
DES-CBC transform 819
encryption 818
ESP authentication data 818
ESP header 818
ESP trailer 818
HMAC-MD5 820
HMAC-SHA-1 820
integrity check 818
Integrity Check Value (ICV) 820
IP fragment 818
IPv6 environment 822
Next Header field 820
Pad Length 820
Padding 820
Payload Data 819
replay protection 819
Security Parameter Index (SPI) 819
Sequence Number 819
transform 810
transport mode 820, 824
tunnel mode 820, 824
encapsulation 812
encryption 777–778, 787, 848
773
encryption algorithm 834
encryption key 778
ephemeral port 10
Ethernet
802.2 Logical Link Control (LLC) 31
ARP 31, 120
DIX 30, 120
DSAP 31
frame formats 30
header fields 30
IEEE 802.3 standard 30, 32
IEEE 802.4 standard 32
IEEE 802.5 standard 32
IEEE 802.x standards 127
IPv6 354
LSAP 31
protocol-type number 30
SNAP 31
SSAP 31
Subnetwork Access Protocol (SNAP) 31
Extended TACACS 873

F

factoring 782
FDDI 33
File Transfer Protocol (FTP)
normal mode 801
passive mode 801
proxy server 800–801

Archived

fingerprint 785
firewall 12, 776, 811, 829
  advanced filtering 798
  application-level gateway 807, 848
  authentication 800
  bastion host 807–808
  circuit-level gateway 803, 846
  demilitarized zone 808
  DMZ 808
  dual-homed gateway 806
  filter rules 797
  FTP normal mode 801
  FTP passive mode 801
  FTP proxy 800–801
  HTTP proxy 800
  IBM Firewall 800
  inbound connections 804
  introduction 12
  logging 800
  non-secure network 795
  outbound connections 804
  packet-filtering 797
  packet-filtering firewall 805
  packet-filtering router 808
  packet-filtering rules 797
  policy 776, 797
  proxy 799, 804, 806
  screened host firewall 807
  screened subnet firewall 808
  secure network 795
  security ID cards 800
  service level filtering 798
  SOCKS 804
  source/destination level filtering 798
  TELNET proxy 801
Flags field 813
For Your Information (FYI) document 25
forwarding capacity 61
Fragment Offset 813
fragmentation 104
fragmentation extension header 817, 822
Frame Relay
  ARP 44
  Data Link Connection Identifier (DLCI) 41
  Network Level Protocol ID (NLPID) 42
  protocol data unit (PDU) 41
  Subnetwork Access Protocol (SNAP) 42
  virtual circuit 41

G

gateway 11–12, 125
gateway address 125

H

hacker 772, 777
hash function 785–786, 789, 840
hash value 785, 788–789
Hashed Message Authentication Code (HMAC) 789
header checksum 813
High Performance Data Transfer (HPDT) 46
HMAC 789
hop limit 332
hop-by-hop extension header 817, 822
HTTP
  proxy 800

I

IAB 22
IASG Coordination Functional Group (ICFG) 62
ICMPv6 352
IDEA 780
IEEE 802.11 27
IEEE 802.x standards 31
IESG 21
impersonation 772
InATMARP 47
initialization vector 792
initialization vector (IV) 779
integrity check 779, 786
integrity checking 773
International Data Encryption Algorithm (IDEA) 780
Internet 13–14
  Acceptable Use Policy (AUP) 16
  Advance Network and Services (ANS) 16
  ANS CO+RE 16
  Commercial Internet Exchange (CIX) 16
  Commercial use of 16
Internet Architecture Board (IAB) 21–22
Internet Assigned Numbers Authority (IANA) 21, 25, 86, 814
Internet Control Message Protocol (ICMP) 109
  Address Mask Reply 117
  Address Mask Request 117
  Destination Unreachable 111
  Echo 111
  Echo Reply 111

Archived

Parameter Problem 115
Path MTU Discovery 109
Ping 117
Redirect 113
Router Advertisement 113
Router Discovery 109
router discovery protocol 114
Router Solicitation 113
Source Quench 112
Time Exceeded 115
Timestamp Reply 116
Timestamp Request 116
Internet Control Message Protocol (ICMPv6) 371
address resolution 354
autoconfiguration 363
DHCP server 364
message format 352
MTU 359
multicast group 365
Multicast Listener Discovery (MLD) 365
multicast listener done message 367
multicast listener report 367
neighbor advertisement message 356
neighbor cache 356
neighbor discovery 353
neighbor solicitation message 354, 364
neighbor unreachability detection (NUD) 363
prefix discovery 357
redirect message 361
router advertisement message 357, 364
router discovery 357
router solicitation message 360, 364
stateful autoconfiguration 363, 371, 380
stateless autoconfiguration 363, 371
tentative address 364
Internet Engineering Steering Group (IESG) 21
Internet Engineering Task Force (IETF) 21–22, 863
internet gateway 11
Internet Group Management Protocol (IGMP) 119, 352
Multicast Listener Discovery (MLD) 365
internet layer 8
Internet Network Information Center (InterNIC) 69
Internet Protocol (IP) 8, 131
datagram 98
datagram header 99
fragmentation 104
IP prefix 97
Loose Source Routing option 105
MTU 104
Record Route option 107
routing options 105
Strict Source Routing option 106
timestamp 107
TTL 101
Internet Service Providers (ISPs) 17
Internet Society (ISOC) 21
Internet2
participants 19
Internet2 mission 18
Internetwork Address Sub-Group (IASG) 62
internetworking 13
internetwork-layer protocol 60
IP 50
protocol stack 125
IP address
exhaustion 86, 329
IP address exhaustion 329
IP datagram 98
introduction 8
IP datagram header 99
IP gateway 11
IP prefix 97
IP Security Architecture (IPSec)
combinations of AH and ESP 823
combined tunnel 826
concepts 810
cryptographic concepts 777
Diffie-Hellman algorithm 784
Diffie-Hellman key exchange 784
Digital Signature Algorithm 790
encapsulation 812
Hashed Message Authentication Code (HMAC) 789
HMAC 789
IPSec module 811
iterated tunneling 823
modulus 783–784
nested tunneling 823
private exponent 784
private key 784
processing sequence 824
public exponent 784
public key 784
RSA algorithm 783
SA bundle 811, 823
Security Association (SA) 810
Security Association Database (SAD) 811

Archived

Security Parameter Index (SPI) 810
Security Policy Database (SPD) 811
transform 810
transport adjacency 823, 826
tunneling 812
IP stack 125
IP Version 6 (IPv6) 352–353, 357, 363
address space 339
anycast address 345
automatic tunneling 381
configured tunneling 385
DNC extensions 368
dual stack 380
extension headers 333
authentication 339, 347
destination options 339
ESP 339
fragment 351
hop-by-hop 335
redid=ipv6ext.type-length-value (TLV) option
format 333
routing 337
type-length-value (TLV) option format 335
flow 346
flow labels 346
format prefix 340
fragment header 339, 351
header checksum 333
header translation 387
Hop-by-Hop header 335
inverse DNS lookups 368
IPv4-compatible address 341
IPv4-mapped address 341
IPv6 addresses 339
link-local unicast address 341
options 333
packet sizes 350
site-local unicast address 341
solicited node multicast address 345
transition from IPv4 379
tunneling 335, 381
automatic 381
configured 385
tunneling over IPv4 networks 381
VPN 863
IP6.INT domain 368
IP-FDDI 33
IPv6 863
IPX 812
ISAKMP/Oakley
application-layer security 839
authentication 783, 839–840
authentication key 837
authentication mechanism 831, 839
authentication method 834
certificate 832, 837, 846
certificate authority 837
certificate payload 837, 839
Certificate Request message 838
certificates 840
composite value 836
cryptographic key 831
cryptographic keys 835, 837, 840
denial-of-service 830
destination port 833
Diffie-Hellman 832, 835–836, 838–840, 842, 845
Diffie-Hellman algorithm 784
digital signature 838–839
Digital Signature Algorithm 790, 837
Digital Signature Standard 831
DOI 833–834
Domain of Interpretation (DOI) 833–834, 842
encryption 839
encryption algorithm 834
Encryption Bit 838
Encryption Flag 841
encryption key 837
exponent 835
Flags field 838
hash function 840
Hash Payload 842–844
identity 835
identity payload 837–839
Identity Protect exchange 832
Initiator Cookie 833–834, 836
ISAKMP header 833–834, 838–839, 844
Key Exchange attribute 840
Key Exchange field 836
Key Exchange Payload 843
KEY_OAKLEY 834–835
keying material 830, 836–837, 844–845
LDAP 838
man-in-the-middle 830
master key 832
master secret 831
Message 1 833, 841, 845
Message 2 834, 843, 845

Message 3 835, 844
Message 4 836
Message 5 837
Message 6 839
Message ID 833–834, 841, 843–844
Message ID field 839
nonce 835–836, 841–842, 844–845
Nonce field 836
Nonce Payload 843
Oakley Main Mode 832, 839
Oakley Quick Mode 840
Perfect Forward Secrecy (PFS) 831, 840
permanent identifier 832, 846
PFS 831, 840
Phase 1 831
Phase 2 831
pre-shared keys 831
private value 835–836
Proposal Payload 833, 835, 842
protection suites 833, 845
PROTO_ISAKMP 833, 835
protocol code point 844
proxy negotiator 841
pseudo-random function 834, 842
public key 831–832
public value 835–836, 838–840, 842, 844–845
remote access 845
remote host 832, 845
Responder Cookie 833–834, 836
revised RSA public key authentication 835
RSA algorithm 783
RSA public key authentication 835
RSA public key encryption 831
secure DNS server 838
secure local cache 838
Security Association 831–832, 835, 839–840
Security Association field 833–834
Security Association Payload 842, 845
Security Payload 843
security protection suite 831
signature payload 837
SKEYID 832, 836, 838, 844
SKEYID_a 837, 842
SKEYID_d 837
SKEYID_e 837
SPI 838, 844–846
Transform Payload 834–835, 842
ISDN
Basic Rate Interface (BRI) 38
B-channel 38
D-channel 38
maximum transmission unit (MTU) 39
NRZ encoding 38
PPP encapsulation 38
Primary Rate Interface (PRI) 38
ISO 13
ISP 829
iterated tunneling 823
ITU-T 13
IV 779

K

KAS 868
KDBM 870
Kerberos Authentication Server (KAS) 868
Kerberos Database Manager (KDBM) 870
Kerberos Key Distribution Server (KKDS) 871
Kerberos System
assumed goals
accounting 864
authentication 864
authorization 864
assumptions 864
authentication process 866, 870
authorization model 871
database management 870
naming 865
instance name 865
principal name 865
realm name 865
key length 782
key management 780, 848, 884
key refresh 773
keyed algorithm 778
key-exchange 780
key-exchange algorithm 783
keying material 830, 836, 844–845
keyspace 778
KKDS 871

L

LAN emulation (LANE) 61
LAN emulation server 58
LAN replacement 50
LAN segment 125
LANE
layer 62

latency 61
Layer 2 Forwarding (L2F) 875
Layer 2 Tunneling Protocol (L2TP) 875
  Access Concentrator 876
  LAC 876
  LNS 876
  NAS 876
  Network Access Server 876
  Network Server 876
  security features 879
  session 876
  tunnel 876
LDAP 838
link layer 8
LIS 48
Logical IP Subnetwork (LIS) 53
long-term key 790
Loose Source Routing 105
Lotus Notes 792
LSAP 31

M

MAC 786, 789
man-in-the-middle 830
man-in-the-middle attack 791, 879
master key 860
master secret 831
maximum transmission unit (MTU) 34, 104
  ICMPv6 359
  ISDN 39
  Path MTU Discovery 350
MD5 788–789
message authentication code 773
message authentication code (MAC) 786, 860
Metropolitan Area Ethernet (MAE) 17
Metropolitan Fiber Systems (MFS) 17
MILNET 14
modular arithmetic 783
MPOA 60–61, 63
  benefits 60
  client 61
  functional group layer 62
  logical components 61
  operation 63
  server 61
multicast 85
  host group 85
  multicast group 365
  Multicast Listener Discovery (MLD) 365
  multicast listener done message 367
  multicast listener report 367
  server (MCS) 63
multicast address resolution server (MARS) 63
multi-homed 77, 97–98
multi-homing 68
Multi-Path Channel+ (MPC+) 46
Multiprotocol Encapsulation 53
mutable fields 813

N

NAS 873
National Institute of Standards and Technology (NIST) 788
National Science Foundation (NSF) 15
National Science Foundation Network (NFSNET) 15
National Security Agency (NSA) 788
neighbor discovery 353
nested tunneling 823
NetBIOS 812
Network Access Points (NAPs) 17
network access server 873
Network Control Program (NCP) 13
network interface layer 8
network layer 8
Next Generation Internet (NGI) initiative 18
NIST 788
nonce 835–836, 841–842, 844–845
non-repudiation 773, 779, 782
NSA 788

O

Oakley Main Mode 839
Oakley Quick Mode 840
one-time password 773
one-way function 785
overlapping fragment attack 813

P

packet-filtering 796
packet-filtering router 796
Path MTU Discovery 109
Perfect Forward Secrecy (PFS) 831, 840
per-session key 791
PFS 831, 840

PGP 780
physical layer 62
Ping 117
point of presence 875
Point-to-Point Protocol (PPP) 874
  authentication 36
  IP Control Protocol (IPCP) 37
  IPCP 37
  L2TP tunnel 877
  LCP 36
  Link Control Protocol (LCP) 36
  NCP 36
  Network Control Protocol (NCP) 36
  Synchronous Digital Hierarchy (SDH) 45
  Synchronous Optical Network (SONET) 45
  Synchronous Payload Envelope (SPE) 46
  Van Jacobson Header Compression 37
Point-to-Point Tunneling Protocol (PPTP) 875
prefix discovery 357
Pretty Good Privacy (PGP) 780
prime factor 790
prime number 782
principal identifier 865
private IP address 812
private key 780, 790
protocol number
  in an IPv6 header 331
protocol virtual LAN (PVLAN) 60
proxy 796
proxy server 846
proxy-ARP 82
  concept 123
pseudo-header
  IPv6 333
pseudo-random function 834
pseudorandom generator 793
public key 780, 790–791, 831, 860
public-key algorithm 782–783
public-key algorithms 781
PVC 48

R

RADIUS 873
random function 793
random-number generator 792
RC2 794
RC4 794
Real-Time Transport Protocol 756
Reconfigure, DHCP 372
Record Route 107
Redbooks Web site 961
  Contact us xxiii
refresh keys 840
remote access 845
remote access server (RAS) 776
Remote Authentication Dial In User Service 873
remote dial-in 872
Remote Forwarder Functional Group (RFFG) 63
remote host 829, 832
replay attack 772
replay protection 815
Request for Comments (RFC)
  Internet Standards Track 22
  purpose 22
  RFC 0791 68
  RFC 0792 109–110
  RFC 0826 47, 51, 119–120
  RFC 0877 39
  RFC 0894 31
  RFC 0903 124
  RFC 0906 126, 129
  RFC 0919 68
  RFC 0922 68, 85
  RFC 0925 123
  RFC 0948 32
  RFC 0950 68, 109–110
  RFC 0951 22, 126, 128
  RFC 1010 32
  RFC 1027 123
  RFC 1034 24
  RFC 1035 24
  RFC 1042 32, 53
  RFC 1055 34
  RFC 1112 119
  RFC 1122 25, 56
  RFC 1123 25
  RFC 1144 35, 37
  RFC 1149 22
  RFC 1166 68–69
  RFC 1191 109, 350
  RFC 1206 25
  RFC 1256 109, 113
  RFC 1325 25
  RFC 1349 68, 100
  RFC 1356 39
  RFC 1437 22
  RFC 1466 89

Archived

RFC 1483 53
RFC 1492 873
RFC 1510 864
RFC 1518 95–96
RFC 1518 - 1520 95
RFC 1519 95
RFC 1520 95, 98
RFC 1542 22, 126
RFC 1577 61
RFC 1579 803
RFC 1594 25
RFC 1618 38
RFC 1619 45
RFC 1661 35
RFC 1662 35
RFC 1700 25, 40, 48
RFC 1755 55
RFC 1809 346
RFC 1812 25
RFC 1827 818
RFC 1886 367–368, 390
RFC 1905 640
RFC 1906 640
RFC 1918 89
RFC 1928 847–848
RFC 1929 847
RFC 1961 847
RFC 2026 21
RFC 2050 89
RFC 2058 873
RFC 2131 130
RFC 2132 126, 129–130, 134, 137
RFC 2138 873
RFC 2181 25
RFC 2185 379
RFC 2223 22
RFC 2225 50, 53
RFC 2236 119
RFC 2246 861
RFC 2341 875
RFC 2362 261
RFC 2373 339, 345, 390
RFC 2374 342
RFC 2375 344
RFC 2400 25, 50
RFC 2402 814, 817
RFC 2406 818, 822
RFC 2427 41
RFC 2460 346, 390
RFC 2461 352, 363, 390
RFC 2462 364–365, 390
RFC 2463 352
RFC 2579 640
RFC 2661 875
RFC 2664 25
RFC 2800 25
RFC 2888 879
RFC 2893 379, 390
RFC 922 85
state 23
  draft standard 23
  experimental 23
  historic 23
  informational 23
  proposed standard 23
  standard 23
status 24
  elective 24
  limited use 24
  not recommended 24
  recommended 24
  required 24
Réseaux IP Européens (RIPE) 86
Resource Reservation Protocol (RSVP) 346
restricted cipher 778
Reverse Address Resolution Protocol (RARP)
  operation code field 125
  packet format 125
  reply 125
  request 125
Route Server Functional Group (RSFG) 63
router 11
  router discovery protocol 114
Router Discovery 109
router discovery 357
router discovery protocol 114
routing 77
  direct 78
  indirect 78
  partial routing information 77
routing extension header 817, 822
routing table 79
routing table explosion 95
RSA 782–783
RSA algorithm 783
RTCP 762
RTP 756
  Control Protocol 762

header format 758

S

SA bundle 811, 826
secret, shared 783
secure DNS server 838
Secure Electronic Transactions (SET) 787
  acquirer 880
  cardholder 880
  certificate authority 881
  issuer 880
  key management 884
  merchant 880
  payment gateway 880
  transactions 881
Secure Hash Algorithm 1 (SHA-1) 788
secure local cache 838
Secure Sockets Layer (SSL) 854
  certificate 859
  change CipherSpec protocol 857
  compatibility 856
  connection state 857
  generate encryption key 860
  handshake phase 855
  Handshake Protocol 854, 858
  master key 860
  message authentication code (MAC) 860
  public key 860
  Record Layer 858
  Record Protocol 854, 860
  security issues 855
  session 856
  session state 857
  states 856
  symmetric-key 860
  TCP port 443 855
Security Association Database (SAD) 811
security exposures 773
security ID cards 800
Security Parameter Index (SPI) 810, 814, 819
security policy 776
Security Policy Database (SPD) 811
security solutions 773, 775
Serial Line IP (SLIP) 874
  addressing
    35
  implementations 35
  overview 34
  Van Jacobson Header Compression 35
SET 792
SHA-1 788
shared keys 782
shared secret 783, 785, 788–789
Simple Internet Transition (SIT) 379
SKEYID 836, 844
SKEYID_a 837, 842
SKEYID_d 837
SKEYID_e 837
SNA
  LCP 36
  Link Control Protocol (LCP) 36
  NCP 36
  Network Control Protocol (NCP) 36
SOCKS
  authentication methods 848
  circuit-level gateway 846
  encapsulation method 850
  encryption standards 848
  firewall 804, 846
  key management systems 848
  method options 849
  request detail message 851
  SOCKS server 846
  SOCKS-enabled client 847
  SOCKS-enabled TCP/IP stack 847
  SOCKSv4 847
  SOCKSv5 847
  TCP connection 849
  tunneling protocols 848
  UDP connection 852
  UDP port 852
  UDP relay server 853
  UDP support 848
  version identifier 849
spoofing attack 879
Standard Protocol Numbers (STD) 24
  STD 01 25, 36
  STD 02 25
  STD 03 25
  STD 04 25
  STD 05 68, 109
  STD 51 35
stream algorithm 779
Strict Source Routing 106
strong cryptography 823
subnet mask 73
  determining 77

Archived

subnet number 73
subnets 72
subnetting
  static 74
  variable length 74
Subnetwork Access Protocol (SNAP) 31, 40, 42
supernetting 96
SVC 48
symmetric algorithm 779–780
symmetric-key 860
Synchronous Digital Hierarchy (SDH) 45
Synchronous Optical Network (SONET) 45
Synchronous Payload Envelope (SPE) 46

T

TACACS 873
TACACS+ 873
tapping the wire 772
TCP
  SOCKS-enabled stack 847
TELNET
  proxy server 801
Terminal Access Controller Access Control System 873
TGS 868
Ticket-Granting Server (TGS) 868
time stamp 782
time-to-live
  IP 101
  IPSec Authentication Header (AH) 813
  IPv6 hop limit 332
Token-Ring LAN 33
transform 810
transparent subnetting 123
transport adjacency 823, 826
triple-DES 780
Trivial File Transfer Protocol (TFTP)
  BOOTP 126
trust chain 792
tunnel 876
tunneling 381, 812, 848
two-way random number handshake 773
Type of Service (TOS) 813
type-length-value (TLV) 335

U

unicast
  address 84
University Corporation for Advanced Internet Development (UCAID) 19

V

value, hash 788
virus 772

W

well-known port 10
Wireless Application Protocol (WAP) 27

X

X.25
  Call Request packet 39
  Call User Data (CUD) 39
  network-layer protocol identifier (NLPID) 39
  Organizationally Unique Identifier (OUI) 41
  Protocol Data Unit (PDU) 39
  Protocol Identifier (PID) 41
  Subnetwork Access Protocol (SNAP) 40
  Subsequent Protocol Identifier (SPI) 39
  virtual circuits 39
X.509 certificates 861
XTACACS 873

Archive

Redbooks

TCP/IP Tutorial and Technical Overview

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

Archive

Archived

Archived



TCP/IP Tutorial and Technical Overview



Redbooks

Understand networking fundamentals of the TCP/IP protocol suite

Introduces advanced concepts and new technologies

Includes the latest TCP/IP protocols

With continual advances in hardware and TCP/IP networking capabilities, this very popular book deserves an update.

The TCP/IP protocol suite has become the de facto standard for computer communications in today's networked world. The ubiquitous implementation of a specific networking standard has led to an incredible dependence on the applications enabled by it. Today, we use the TCP/IP protocols and the Internet not only for entertainment and information, but to conduct our business by performing transactions, buying and selling products, and delivering services to customers. We are continually extending the set of applications that leverage TCP/IP, thereby driving the need for further infrastructure support.

It is our hope that both the novice and the expert will find useful information in this publication.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

GG24-3376-07 ISBN 0738494682