

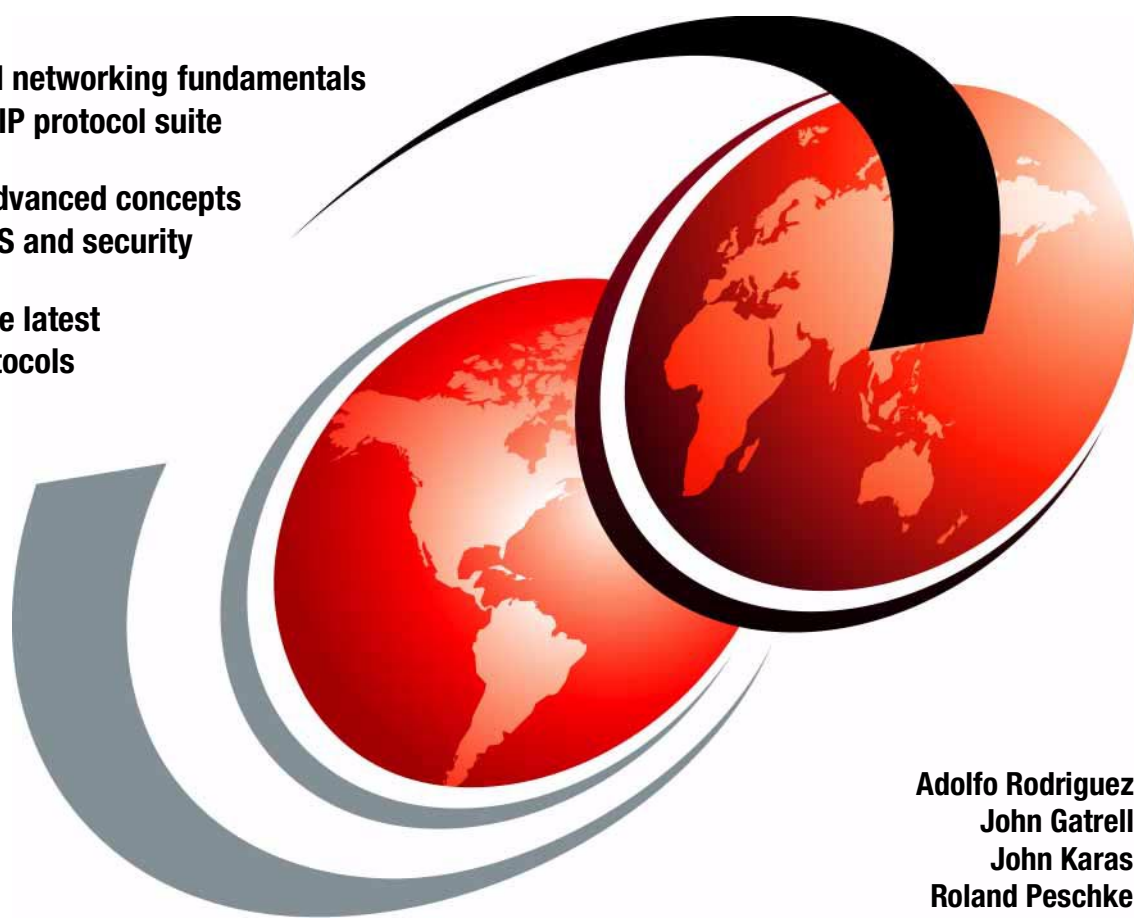


TCP/IP Tutorial and Technical Overview

Understand networking fundamentals of the TCP/IP protocol suite

Contains advanced concepts such as QoS and security

Includes the latest TCP/IP protocols



Adolfo Rodriguez
John Gatrell
John Karas
Roland Peschke

ibm.com/redbooks

Redbooks

GG24-3376-06



International Technical Support Organization

TCP/IP Tutorial and Technical Overview

August 2001

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, "Special notices" on page 913.

Seventh Edition (August 2001)

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1989-2001. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	xix
The team that wrote this redbook	xix
Comments welcome	xxi

Part 1. Core TCP/IP protocols 1

Chapter 1. Architecture, history, standards, and trends	3
1.1 TCP/IP architectural model	3
1.1.1 Internetworking	3
1.1.2 The TCP/IP protocol layers	5
1.1.3 TCP/IP applications	8
1.1.4 Bridges, routers, and gateways	10
1.2 The roots of the Internet	11
1.2.1 ARPANET	13
1.2.2 NSFNET	14
1.2.3 Commercial use of the Internet	16
1.2.4 Internet2	17
1.2.5 The Open Systems Interconnection (OSI) Reference Model	19
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
Chapter 2. Network interfaces	29
2.1 Ethernet and IEEE 802.x Local Area Networks (LANs)	29
2.2 Fiber Distributed Data Interface (FDDI)	32
2.3 Serial Line IP (SLIP)	33
2.4 Point-to-Point Protocol (PPP)	34
2.4.1 Point-to-Point encapsulation	35
2.5 Integrated Services Digital Network (ISDN)	36
2.6 X.25	38
2.7 Frame relay	40
2.7.1 Frame format	40
2.7.2 Interconnect issues	41
2.7.3 Data link layer parameter negotiation	41
2.7.4 IP over frame relay	42
2.8 PPP over SONET and SDH circuits	43

2.8.1 Physical layer	43
2.9 Multi-Path Channel+ (MPC+)	44
2.10 Asynchronous Transfer Mode (ATM)	44
2.10.1 Address resolution (ATMARP and InATMARP)	45
2.10.2 Classical IP over ATM	48
2.10.3 ATM LAN emulation	54
2.10.4 Classical IP over ATM versus LAN emulation	57
2.11 Multiprotocol over ATM (MPOA)	58
2.11.1 Benefits of MPOA	58
2.11.2 MPOA logical components	59
2.11.3 MPOA functional components	59
2.11.4 MPOA operation	62
2.12 References	63
Chapter 3. Internetworking protocols	65
3.1 Internet Protocol (IP)	65
3.1.1 IP addressing	65
3.1.2 IP subnets	69
3.1.3 IP routing	74
3.1.4 Methods of delivery - unicast, broadcast, multicast, and anycast	80
3.1.5 The IP address exhaustion problem	83
3.1.6 Intranets - private IP addresses	86
3.1.7 Classless Inter-Domain Routing (CIDR)	86
3.1.8 IP datagram	90
3.2 Internet Control Message Protocol (ICMP)	102
3.2.1 ICMP messages	103
3.2.2 ICMP applications	112
3.3 Internet Group Management Protocol (IGMP)	113
3.4 Address Resolution Protocol (ARP)	114
3.4.1 ARP overview	114
3.4.2 ARP detailed concept	115
3.4.3 ARP and subnets	118
3.4.4 Proxy-ARP or transparent subnetting	118
3.5 Reverse Address Resolution Protocol (RARP)	120
3.5.1 RARP concept	120
3.6 Bootstrap protocol (BOOTP)	121
3.6.1 BOOTP forwarding	125
3.6.2 BOOTP considerations	126
3.7 Dynamic Host Configuration Protocol (DHCP)	126
3.7.1 The DHCP message format	127
3.7.2 DHCP message types	129
3.7.3 Allocating a new network address	130
3.7.4 DHCP lease renewal process	133

3.7.5 Reusing a previously allocated network address.	134
3.7.6 Configuration parameters repository.	135
3.7.7 DHCP considerations	135
3.7.8 BOOTP and DHCP interoperability	136
Chapter 4. Routing protocols	137
4.1 Autonomous systems	138
4.2 Types of IP routing and IP routing algorithms	140
4.2.1 Static routing	140
4.2.2 Distance vector routing.	142
4.2.3 Link state routing	143
4.2.4 Hybrid routing.	144
4.3 Routing Information Protocol (RIP).	145
4.3.1 RIP packet types	145
4.3.2 RIP packet format.	146
4.3.3 RIP modes of operation	146
4.3.4 Calculating distance vectors	147
4.3.5 Convergence and counting to infinity	148
4.3.6 RIP limitations	152
4.4 Routing Information Protocol Version 2 (RIP-2)	152
4.4.1 RIP-2 packet format	153
4.4.2 RIP-2 limitations.	155
4.5 RIPng for IPv6	155
4.5.1 Differences between RIPng and RIP-2	155
4.5.2 RIPng packet format.	156
4.6 Open Shortest Path First (OSPF)	158
4.6.1 OSPF terminology	159
4.6.2 OSPF packet types.	165
4.6.3 Neighbor communication	166
4.6.4 OSPF neighbor state machine	168
4.6.5 OSPF virtual links and transit areas	169
4.6.6 OSPF route redistribution	170
4.6.7 OSPF stub areas	172
4.6.8 OSPF route summarization.	173
4.7 Enhanced Interior Gateway Routing Protocol (EIGRP)	174
4.7.1 Features of EIGRP	174
4.7.2 Terminology	175
4.7.3 Neighbor discovery and recovery	177
4.7.4 The DUAL algorithm	177
4.7.5 EIGRP packet types	179
4.8 Exterior Gateway Protocol (EGP)	180
4.9 Border Gateway Protocol (BGP)	180
4.9.1 BGP concepts and terminology.	181

4.9.2	IBGP and EBGP communication	183
4.9.3	Protocol description	185
4.9.4	Path selection	188
4.9.5	BGP synchronization	191
4.9.6	BGP aggregation	193
4.9.7	BGP confederations	194
4.9.8	BGP route reflectors	196
4.10	Routing protocol selection	197
4.11	Additional functions performed by the router	198
4.12	Routing processes in UNIX-based systems	199
Chapter 5. Transport layer protocols		201
5.1	Ports and sockets	201
5.1.1	Ports	201
5.1.2	Sockets	202
5.2	User Datagram Protocol (UDP)	204
5.2.1	UDP datagram format	205
5.2.2	UDP application programming interface	206
5.3	Transmission Control Protocol (TCP)	206
5.3.1	TCP concept	207
5.3.2	TCP application programming interface	220
5.3.3	TCP congestion control algorithms	221
Chapter 6. IP multicast		229
6.1	Multicast addressing	229
6.1.1	Multicasting on a single physical network	230
6.1.2	Multicasting between network segments	231
6.2	Internet Group Management Protocol (IGMP)	232
6.2.1	IGMP messages	233
6.2.2	IGMP operation	234
6.3	Multicast delivery tree	235
6.4	Multicast forwarding algorithms	236
6.4.1	Reverse path forwarding algorithm	236
6.4.2	Center-based tree algorithm	237
6.4.3	Multicast routing protocols	238
6.5	Distance Vector Multicast Routing Protocol (DVMRP)	238
6.5.1	Protocol overview	239
6.5.2	Building and maintaining multicast delivery trees	240
6.5.3	DVMRP tunnels	242
6.6	Multicast OSPF (MOSPF)	242
6.6.1	Protocol overview	243
6.6.2	MOSPF and multiple OSPF areas	244
6.6.3	MOSPF and multiple autonomous systems	245

6.6.4	MOSPF interoperability.	245
6.7	Protocol Independent Multicast (PIM).	245
6.7.1	PIM dense mode	246
6.7.2	PIM sparse mode	247
6.8	Interconnecting multicast domains	251
6.8.1	Multicast Source Discovery Protocol (MSDP)	251
6.8.2	Border Gateway Multicast Protocol.	253
6.9	The multicast backbone	254
6.9.1	MBONE routing	254
6.9.2	Multicast applications	256
Part 2.	TCP/IP application protocols	259
 Chapter 7. Application structure and programming interfaces		261
7.1	Characteristics of applications	261
7.1.1	Client/server model.	261
7.2	Application programming interfaces (APIs).	262
7.2.1	The socket API	262
7.2.2	Remote Procedure Call (RPC)	266
7.2.3	Windows Sockets Version 2 (Winsock V2.0).	271
7.2.4	SNMP Distributed Programming Interface (SNMP DPI)	273
7.2.5	FTP API	276
7.2.6	CICS socket interface.	277
7.2.7	IMS socket interface	277
7.2.8	Sockets Extended.	277
7.2.9	REXX sockets	278
 Chapter 8. Directory and naming protocols		279
8.1	Domain Name System (DNS).	279
8.1.1	The hierarchical namespace.	280
8.1.2	Fully Qualified Domain Names (FQDNs).	281
8.1.3	Generic domains	281
8.1.4	Country domains	282
8.1.5	Mapping domain names to IP addresses	282
8.1.6	Mapping IP addresses to domain names – pointer queries	282
8.1.7	The distributed name space	283
8.1.8	Domain name resolution.	284
8.1.9	Domain Name System resource records.	288
8.1.10	Domain Name System messages	290
8.1.11	A simple scenario	294
8.1.12	Extended scenario	297
8.1.13	Transport	298
8.1.14	DNS applications	299

8.1.15	References	299
8.2	Dynamic Domain Name System	300
8.2.1	The UPDATE DNS message format	301
8.2.2	The IBM implementation of DDNS	304
8.2.3	Proxy A Record update (ProxyArec)	313
8.3	Network Information System (NIS)	315
8.4	Lightweight Directory Access Protocol (LDAP)	316
8.4.1	LDAP - lightweight access to X.500	317
8.4.2	The LDAP directory server	319
8.4.3	Overview of LDAP architecture	320
8.4.4	LDAP models	321
8.4.5	LDAP security	329
8.4.6	LDAP URLs	331
8.4.7	LDAP and DCE	332
8.4.8	The directory-enabled networks initiative (DEN)	334
8.4.9	Web-Based Enterprise Management (WBEM)	335
8.4.10	References	335
Chapter 9. Remote execution and distributed computing		337
9.1	TELNET	337
9.1.1	TELNET operation	337
9.1.2	Terminal emulation (Telnet 3270)	344
9.1.3	TN3270 enhancements (TN3270E)	345
9.1.4	References	347
9.2	Remote Execution Command protocol (REXEC and RSH)	347
9.2.1	Principle of operation	348
9.3	Introduction to the Distributed Computing Environment (DCE)	348
9.3.1	DCE directory service	350
9.3.2	DCE security service	352
9.3.3	DCE threads	357
9.3.4	DCE remote procedure call	358
9.3.5	Distributed time service	359
9.3.6	Distributed file service (DFS)	361
9.3.7	References	364
Chapter 10. File related protocols		365
10.1	File Transfer Protocol (FTP)	365
10.1.1	Overview of FTP	365
10.1.2	FTP operations	366
10.1.3	Reply codes	369
10.1.4	FTP scenario	370
10.1.5	A sample FTP session	371
10.1.6	Anonymous FTP	371

10.1.7 Remote job entry using FTP	371
10.2 Trivial File Transfer Protocol (TFTP).	371
10.2.1 TFTP usage	372
10.2.2 Protocol description	372
10.2.3 TFTP multicast option.	375
10.2.4 Security issues	375
10.3 Network File System (NFS)	375
10.3.1 NFS concept.	376
10.3.2 NFS Version 4	381
10.3.3 WebNFS.	382
10.3.4 References.	383
10.4 The Andrew File System (AFS)	383
Chapter 11. Mail applications	387
11.1 Simple Mail Transfer Protocol (SMTP)	387
11.1.1 How SMTP works	389
11.1.2 SMTP and the Domain Name System.	396
11.1.3 References.	398
11.2 Multipurpose Internet Mail Extensions (MIME)	399
11.2.1 How MIME works	402
11.2.2 The Content-Type field	403
11.2.3 The Content-Transfer-Encoding field	410
11.2.4 Using non-ASCII characters in message headers.	416
11.2.5 References.	418
11.3 Post Office Protocol (POP).	418
11.3.1 POP3 commands and responses	419
11.3.2 References.	420
11.4 Internet Message Access Protocol Version 4 (IMAP4)	420
11.4.1 IMAP4 underlying electronic mail models	421
11.4.2 IMAP4 commands and responses.	421
11.4.3 Message numbers	422
11.4.4 IMAP4 states	423
11.4.5 Client commands	425
11.4.6 References.	426
Chapter 12. The World Wide Web	427
12.1 Web browsers	427
12.2 Web servers.	429
12.3 Hypertext Transfer Protocol (HTTP)	429
12.3.1 Overview of HTTP	430
12.3.2 HTTP operation	431
12.4 Content	440
12.4.1 Static content	441

12.4.2	Client-side dynamic content	441
12.4.3	Server-side dynamic content	442
12.4.4	Objects	443
12.4.5	Developing content with IBM Web Application Servers	446
12.5	References	447
Chapter 13.	Multimedia protocols	449
13.1	Real-Time Protocols: RTP and RTCP	449
13.1.1	The Real-Time Transport Protocol (RTP)	450
13.1.2	The Real-Time Control Protocol	455
13.1.3	RTCP packet format	457
13.1.4	RTP translators and mixers	459
13.1.5	Real-time applications	462
13.2	IP telephony	463
13.2.1	Introduction	463
13.2.2	The IP telephony protocol stack	464
13.2.3	ITU-T recommendation H.323	465
13.2.4	Session Initiation Protocol (SIP)	470
13.2.5	Media Gateway Control Protocol (MGCP)	472
13.2.6	Media Gateway Controller (Megaco)	473
13.2.7	Signaling protocol functional comparison	474
13.2.8	Voice encoding and compression	476
Chapter 14.	Wireless Application Protocol (WAP)	479
14.1	The WAP environment	479
14.2	Key elements of the WAP specifications	480
14.2.1	Overview of the WAP programming model	480
14.2.2	WAP network configurations	483
14.3	Wireless Markup Language (WML) and WMLScript	486
14.3.1	WML	486
14.3.2	WMLScript	488
14.4	Push architecture	489
14.4.1	Push framework	490
14.4.2	Push proxy gateway (PPG)	491
14.4.3	Push access control protocol (PAP)	492
14.4.4	Service indication	493
14.4.5	Push over-the-air protocol (OTA)	494
14.4.6	Client-side infrastructure	494
14.4.7	Security	494
14.5	Overview of the WAP protocol stack	495
14.5.1	Wireless application environment (WAE)	496
14.5.2	Wireless Telephony Application (WTA)	498
14.5.3	Wireless Session Protocol (WSP)	498

14.5.4	Wireless Transaction Protocol (WTP)	511
14.5.5	Wireless Transport Layer Security (WTLS)	514
14.5.6	Wireless Datagram Protocol (WDP)	519
14.6	Protocol summary	521
Chapter 15.	Network management	525
15.1	Simple Network Management Protocol and MIB overview	525
15.2	Structure and identification of management information (SMI)	526
15.3	Management Information Base (MIB)	528
15.3.1	IBM-specific MIB part	531
15.4	Simple Network Management Protocol (SNMP)	532
15.5	Simple Network Management Protocol Version 2 (SNMPv2)	535
15.5.1	SNMPv2 entity	536
15.5.2	SNMPv2 party	536
15.5.3	GetBulkRequest	537
15.5.4	InformRequest	539
15.6	MIB for SNMPv2.	539
15.7	The new administrative model	540
15.8	Simple Network Management Protocol Version 3 (SNMPv3)	542
15.8.1	Single authentication and privacy protocol	543
15.9	References.	544
Chapter 16.	Utilities	547
16.1	Remote printing (LPR and LPD)	547
16.2	X Window system.	547
16.2.1	Functional concept	548
16.2.2	Protocol	553
16.3	Network News Transfer Protocol (NNTP)	553
16.4	Finger protocol	554
16.5	Netstat	554
Part 3. Advanced concepts and new technologies.		557
Chapter 17.	IP Version 6.	559
17.1	IPv6 overview.	560
17.2	The IPv6 header format	561
17.2.1	Packet sizes	564
17.2.2	Extension headers	565
17.2.3	IPv6 addressing	572
17.2.4	Traffic class	578
17.2.5	Flow labels	579
17.3	Internet Control Message Protocol Version 6 (ICMPv6)	579
17.3.1	Neighbor discovery.	581

17.3.2 Stateless address autoconfiguration	590
17.3.3 Multicast Listener Discovery (MLD)	592
17.4 DNS in IPv6	595
17.4.1 Format of IPv6 resource records	595
17.5 DHCP in IPv6	598
17.5.1 Differences between DHCPv6 and DHCPv4	599
17.5.2 DHCPv6 messages	599
17.6 Mobility support in IPv6	600
17.7 Internet transition - Migrating from IPv4 to IPv6	601
17.7.1 Dual IP stack implementation - the IPv6/IPv4 node	601
17.7.2 Tunneling	603
17.7.3 Header translation	610
17.7.4 Interoperability summary	610
17.8 The drive towards IPv6	611
17.9 References	612
Chapter 18. Multiprotocol Label Switching (MPLS)	613
18.1 MPLS overview	613
18.1.1 Conventional routing model	613
18.1.2 MPLS forwarding model	613
18.1.3 Additional benefits	614
18.2 Components of an MPLS network	615
18.2.1 Terminology	616
18.2.2 Label swapping	618
18.2.3 Label switched path (LSP)	620
18.2.4 Label stack and label hierarchies	620
18.2.5 MPLS stacks in a BGP environment	622
18.3 Label distribution protocols	624
18.3.1 Types of label distribution protocols	624
18.3.2 Label distribution methods	625
18.4 Stream merge	625
18.4.1 Merging in a frame-based environment	625
18.4.2 Merging in an ATM environment	626
18.5 Multiprotocol Lambda Switching	627
Chapter 19. Mobile IP	629
19.1 Mobile IP overview	629
19.2 Mobile IP operation	630
19.3 Mobility agent advertisement extensions	632
19.4 Mobile IP registration process	634
19.5 Tunneling	637
19.6 Broadcast datagrams	638
19.7 Move detection	638

19.7.1	Returning home	639
19.8	ARP considerations	639
19.9	Mobile IP security considerations	639
Chapter 20.	Integrating other protocols with TCP/IP	641
20.1	Enterprise Extender	641
20.1.1	Performance and recovery	642
20.2	Data Link Switching	642
20.2.1	Introduction	642
20.2.2	Functional description.	643
20.3	Multiprotocol Transport Network (MPTN)	645
20.3.1	Requirements for mixed-protocol networking	645
20.3.2	MPTN architecture	646
20.3.3	MPTN methodology	646
20.3.4	MPTN major components	647
20.4	NetBIOS over TCP/IP.	649
20.4.1	NetBIOS Name Server (NBNS) implementations	652
Chapter 21.	TCP/IP security.	655
21.1	Security exposures and solutions	655
21.1.1	Common attacks against security	655
21.1.2	Solutions to network security problems.	656
21.1.3	Implementations of security solutions	657
21.1.4	Network security policy	659
21.2	A short introduction to cryptography	660
21.2.1	Terminology	660
21.2.2	Symmetric or secret-key algorithms	663
21.2.3	Asymmetric or public-key algorithms.	664
21.2.4	Hash functions	669
21.2.5	Digital certificates and certification authorities	675
21.2.6	Random-number generators	676
21.2.7	Export/import restrictions on cryptography	677
21.3	Firewalls.	678
21.3.1	Firewall concept	679
21.3.2	Components of a firewall system	680
21.3.3	Packet-filtering router	680
21.3.4	Application level gateway (proxy)	682
21.3.5	Circuit level gateway.	687
21.3.6	Types of firewall	689
21.4	Network Address Translation (NAT)	694
21.4.1	NAT concept.	694
21.4.2	Translation mechanism.	695
21.4.3	NAT limitations	697

21.5 The IP security architecture (IPsec)	698
21.5.1 Concepts	698
21.5.2 Authentication Header (AH)	702
21.5.3 Encapsulating Security Payload (ESP)	708
21.5.4 Combining IPsec protocols	715
21.5.5 The Internet Key Exchange protocol (IKE)	721
21.6 SOCKS	739
21.6.1 SOCKS Version 5 (SOCKSv5)	741
21.7 Secure Shell (I)	746
21.7.1 SSH overview	747
21.8 Secure Sockets Layer (SSL)	747
21.8.1 SSL overview	747
21.8.2 SSL protocol	749
21.9 Transport Layer Security (TLS)	755
21.10 Secure Multipurpose Internet Mail Extension (S-MIME)	755
21.11 Virtual private networks (VPN) overview	755
21.11.1 VPN Introduction and benefits	755
21.12 Kerberos authentication and authorization system	757
21.12.1 Assumptions	758
21.12.2 Naming	758
21.12.3 Kerberos authentication process	759
21.12.4 Kerberos database management	763
21.12.5 Kerberos Authorization Model	764
21.12.6 Kerberos Version 5 enhancements	764
21.13 Remote access authentication protocols	765
21.14 Layer 2 Tunneling Protocol (L2TP)	768
21.14.1 Terminology	768
21.14.2 Protocol overview	769
21.14.3 L2TP security issues	772
21.15 Secure electronic transactions (SET)	773
21.15.1 SET roles	773
21.15.2 SET transactions	774
21.15.3 The SET certificate scheme	777
21.16 References	778
Chapter 22. Quality of Service	781
22.1 Why QoS?	781
22.2 Integrated Services	782
22.2.1 Service classes	786
22.2.2 The Resource Reservation Protocol (RSVP)	790
22.2.3 Integrated Services outlook	803
22.3 Differentiated Services	804
22.3.1 Differentiated Services architecture	806

22.3.2	Integrated Services (Intserv) over Diffserv networks	815
22.3.3	Configuration and administration of DS with LDAP	818
22.3.4	Using Differentiated Services with IPSec	819
22.4	References	820
Chapter 23.	Availability, scalability, and load balancing	823
23.1	Availability	823
23.2	Scalability	825
23.3	Load balancing	825
23.4	Terms used in this chapter	826
23.4.1	Sysplex	826
23.4.2	Workload Manager (WLM)	826
23.4.3	Virtual IP-address (VIPA)	827
23.4.4	Dynamic XCF	828
23.4.5	Dynamic IP addressing in a sysplex	828
23.4.6	Takeover/takeback of DVIPA addresses	830
23.5	Introduction of available solutions	832
23.6	Network Dispatcher	833
23.6.1	Network Dispatcher components	833
23.6.2	Load balancing with weights	837
23.6.3	High availability	838
23.6.4	Server affinity	840
23.6.5	Rules-based balancing	840
23.6.6	Wide Area Network Dispatcher	840
23.6.7	Combining ISS and Dispatcher	841
23.6.8	Advisors and custom advisors	842
23.6.9	SNMP support	843
23.6.10	Co-location option	843
23.6.11	ISP configuration	844
23.6.12	OS/390 Parallel Sysplex support	845
23.7	Cisco LocalDirector	847
23.7.1	Overview	847
23.7.2	Connection and datagram flow	848
23.8	IBM Sysplex Distributor	849
23.8.1	Sysplex Distributor elements	849
23.8.2	Sysplex Distributor initialization and takeover/takeback	850
23.8.3	Sysplex Distributor load balancing rules	851
23.8.4	Handling connection requests	851
23.8.5	Data path after connection establishment	851
23.8.6	Takeover/takeback	852
23.8.7	Attaining availability, scalability, and load balancing	853
23.9	Cisco MultiNode Load Balancing (MNLB)	854
23.9.1	Overview of the MultiNode Load Balancing functions	855

23.9.2	Connection establishment and subsequent data flow	856
23.9.3	Client-server connection restart	859
23.9.4	Attaining availability, scalability, and load balancing	859
23.10	IBM Sysplex Distributor and Cisco MNLB	861
23.10.1	What does this mean?	862
23.10.2	Overview of IBM Sysplex Distributor with Service Manager	863
23.10.3	Cisco Forwarding Agent: overview and functions	864
23.10.4	Cisco Workload Agent	864
23.10.5	Connection establishment process	864
23.10.6	Stack, Server, or LPAR failure	867
23.10.7	Failure of the Sysplex Distributor	867
23.10.8	Routing packets	867
23.10.9	Additional tasks of the MNLB components	868
23.11	OS/390 DNS/WLM	870
23.11.1	DNS in a sysplex environment	870
23.11.2	DNS/WLM with remote name server	874
23.12	Virtual Router Redundancy Protocol (VRRP)	875
23.12.1	Introduction	876
23.12.2	VRRP Definitions	877
23.12.3	VRRP overview	877
23.12.4	Sample configuration	879
23.12.5	VRRP packet format	880
23.13	Round-robin DNS	883
23.14	Alternative solutions to load balancing	884
23.14.1	Network address translation	884
23.14.2	Encapsulation	887
Appendix A. Platform implementations		889
A.1	IBM Communications Server for OS/390 V2R10	889
A.1.1	Supported connectivity protocols and devices	889
A.1.2	Supported routing applications	892
A.1.3	Enterprise Extender	893
A.1.4	Virtual IP Addressing (VIPA)	893
A.1.5	Sysplex Distributor	894
A.1.6	Quality of Service (QoS)	896
A.2	IBM OS/400 V5R1	897
A.2.1	GUI configuration support	897
A.2.2	TCP/IP Connectivity Utilities for IBM @server iSeries	898
A.2.3	Dynamic IP routing (RIP and RIP2)	898
A.2.4	Advanced functions	899
A.2.5	Proxy Address Resolution Protocol (Proxy ARP)	900
A.2.6	Point-to-Point Protocol (PPP)	901
A.2.7	Security features	901

A.2.8 Virtual IP Addressing (VIPA)	903
A.2.9 Application programming interfaces (APIs)	903
A.2.10 Supported applications	905
A.3 Linux	909
A.3.1 Linux firewall	909
A.4 The network computer	910
Appendix B. Special notices	913
Appendix C. Related publications	917
C.1 IBM Redbooks	917
C.2 IBM Redbooks collections	918
C.3 Other resources	918
C.4 Referenced Web sites	920
How to get IBM Redbooks	921
IBM Redbooks fax order form	922
Abbreviations and acronyms	923
Index	931
IBM Redbooks review	957

Preface

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 infrastructural support.

In TCP/IP Tutorial and Technical Overview, we take an in-depth look into the TCP/IP protocol suite. In Part I, we introduce TCP/IP, providing a basic understanding of the underlying concepts essential to the protocols. We continue our discussion in Part II with a survey of today's most popular TCP/IP application protocols, including emerging wireless and multimedia applications.

Finally, in Part III, we cover advanced concepts and the latest infrastructural trends in networking, including IPv6, security, Quality of Service, IP mobility, and MPLS. We address the challenges that TCP/IP is currently facing and the technology being developed to overcome them.

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 Raleigh Center.

Adolfo Rodriguez is an Advisory I/T Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of TCP/IP. Before joining the ITSO, Adolfo worked in the design and development of Communications Server for OS/390, in RTP, NC. He holds a B.A. degree in Mathematics and B.S. and M.S. degrees in Computer Science from Duke University, Durham, NC. He is currently pursuing a Ph.D. degree in Computer Science at Duke University, with a concentration on Networking Systems.

John Gatrell works for IBM in the UK. He has 15 years experience in communications customer support, and a further seven years in programming. He holds a B.A. Honours degree in Physics from Oxford University. His specialized areas include UNIX and communications.

John Karas is a network architect in IBM Global Services in the United States. He has 14 years of experience in the data networking field. He holds a Masters of Science degree in Telecommunications from Pace University. His areas of expertise include IP routing algorithms, complex network design, capacity planning, and application performance testing. He has written extensively on supporting OSPF and BGP networks, as well as performance monitoring in SAP environments.

Roland Peschke is a Senior IT Networking Specialist working for IBM customers requesting consulting and education services for the OS/390 TCP/IP and SNA environment. His comprehensive experiences in these areas come from working at IBM Germany and ITSO Raleigh for more than three decades. He worked intensively on several SNA- and TCP/IP Redbooks.

Thanks to the following people for their invaluable contributions to this project:

International Technical Support Organization, Raleigh Center

Gail Christensen, Margaret Ticknor, Jeanne Tucker, David Watts, Juan Rodriguez, Byron Braswell, Thomas Barlen, Linda Robinson

International Technical Support Organization, Austin Center

Wade Wallace and Matthew Parente

IBM Communication Server for OS/390 Development

Jeff Haggar, Bebe Isrel, Dinakaran Joseph

Cisco Systems, Inc.

Rick Williams and Edward Mazurek

BOPS, Inc.

Ricardo Rodriguez

North Carolina State University

Karina Rodriguez

First Edition authors

Peter Frick, Gerard Bourbigot, Frank Vandewiele

Second Edition authors

Peter Frick, Lesia Cox, Ricardo Haragutchi

Third Edition authors

Philippe Beaupied and Frederic Debulois

Fourth Edition authors

Philippe Beaupied and Francis Li

Fifth Edition authors

Eamon Murphy, Matthias Enders, Steve Hayes

Sixth Edition authors

Martin Murhammer, Orcun Atakan, Stefan Bretz, Larry Pugh, Kazunari Suzuki, David Wood

Comments welcome**Your comments are important to us!**

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

- Fax the evaluation form found in “IBM Redbooks review” on page 957 to the fax number shown on the form.
- Use the online evaluation form found at ibm.com/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Core TCP/IP protocols

2 TCP/IP Tutorial and Technical Overview

Chapter 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 lead 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.

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. We use the more common, shorter term, TCP/IP, to refer to the entire protocol suite in this book.

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 is 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. Hence, 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. Currently the backbones are NSFNET in the US, EBONE in Europe, and large commercial backbones.
- **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. Hence, 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 shows two examples of internets. Each is comprised of two or more physical networks.

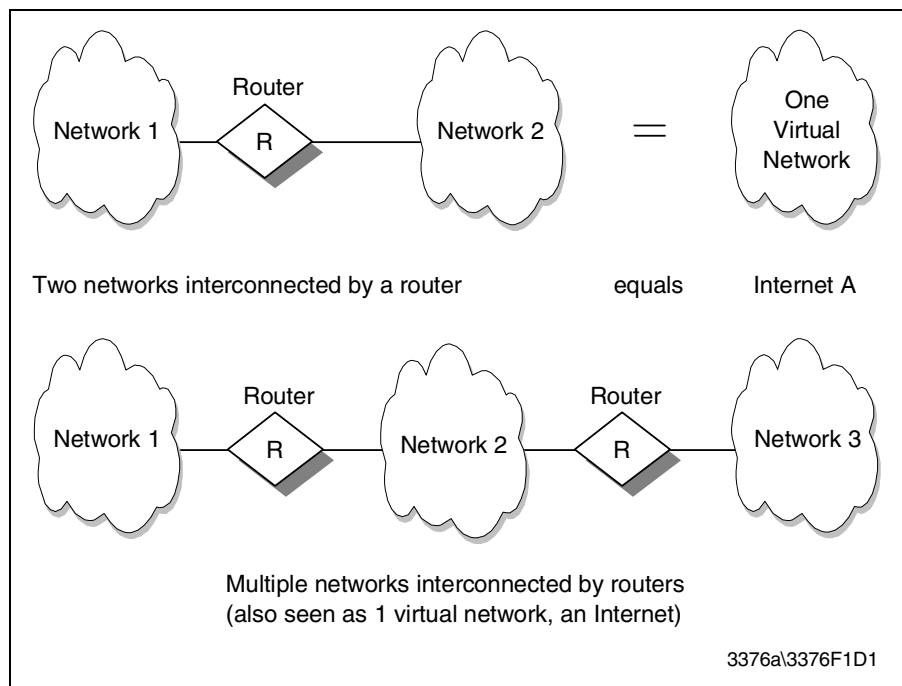


Figure 1. Internet examples - Two interconnected sets of networks, each seen as one logical network

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, 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 5).

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. The addressing scheme is described in detail in 3.1.1, “IP addressing” on page 65.

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, as 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 2 shows how the TCP/IP protocols are modeled in four layers.

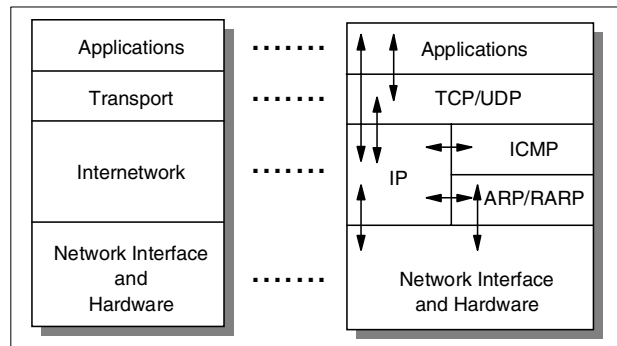


Figure 2. The TCP/IP protocol stack - Each layer represents a package of functions

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 is described in more detail in 5.1, “Ports and sockets” on page 201.

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. It is discussed in more detail in 5.3, “Transmission Control Protocol (TCP)” on

page 206.

Another transport layer protocol is the User Datagram Protocol (UDP, discussed in 5.2, “User Datagram Protocol (UDP)” on page 204). 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 it is so 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 doesn't 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. IP is discussed in detail in 3.1, “Internet Protocol (IP)” on page 65. 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. Some physical networks and interfaces are discussed 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 3.

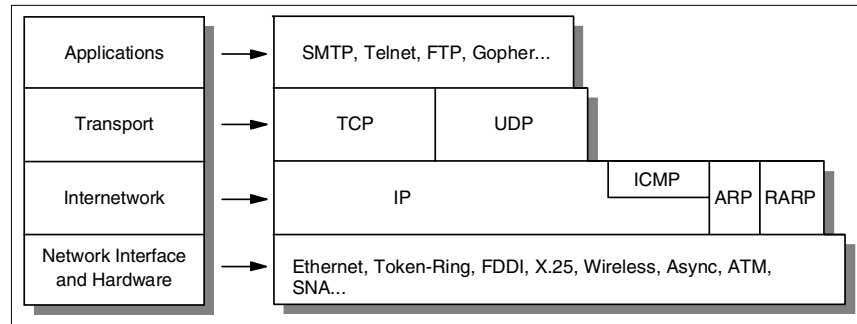


Figure 3. Detailed architectural model

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.
 - FTP (file transfer protocol) for high-speed disk-to-disk file transfers.
 - SMTP (simple mail transfer protocol) 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 reduced protocol overhead.

- Most applications use the client/server model of interaction.

1.1.3.1 The client/server model

TCP is a peer-to-peer, connection-oriented protocol. There are no master/slave relationships. The applications, however, typically use a client/server model for communications.

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.

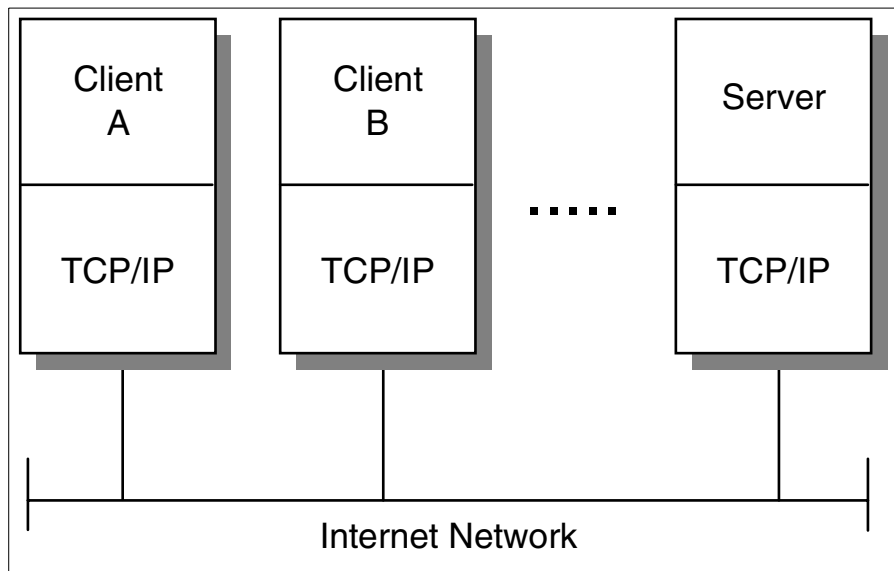


Figure 4. The client/server model of applications

Most servers wait for requests at a *well-known port* so that their clients know 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 wish 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 on TCP/IP application protocols, please refer to Part 2, “TCP/IP application protocols” on page 259.

1.1.4 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 may 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 may 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 21.3, "Firewalls" on page 678 for more information on 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 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 60s and 70s, 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 wishes 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 70s 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*. Some examples of the different networks that have played key roles in this development are described 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 60s 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 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), whereas 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 (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 percent 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* (NSS), supplied by IBM. Each NSS was composed of nine IBM RISC systems (running an IBM version of 4.3BSD UNIX) loosely coupled via 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 (45Mbps) was completed in late 1992. The subsequent migration to gigabit levels has already started and is continuing today.

In April 1995, the US 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/or 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 80s 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 (96 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.

1.2.4.1 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.

The goals of Internet2 are the following:

- 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, etc.) 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 applications 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.

1.2.4.2 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 on Internet2, see their Web page at:

<http://www.internet2.edu>

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 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.

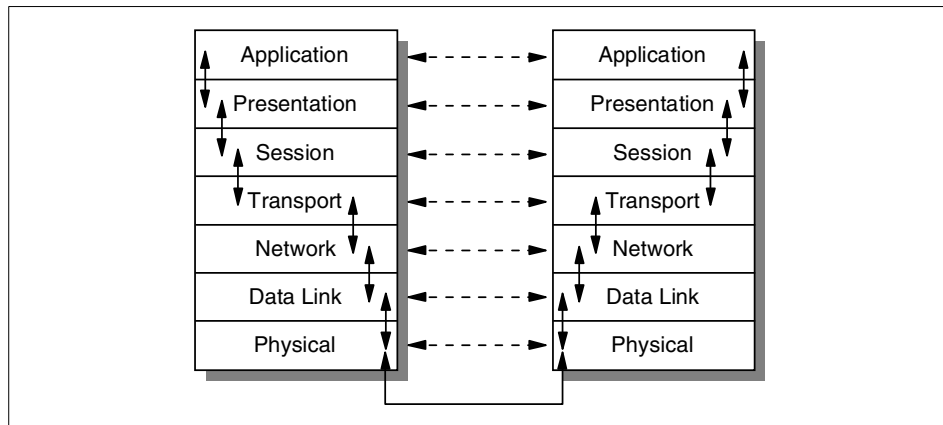


Figure 5. The OSI Reference Model

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 RFCs, 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. On the other hand, this openness could easily turn into a sword with two edges 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 proposed.

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 below:

- 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.

- Once 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 Comment (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, of 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 may 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.

Once an RFC has been published, all revisions and replacements are published as new RFCs. A new RFC which 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 whether two people are referring to different versions of an RFC, since there is never more than one current version.

¹ Some of these protocols, particularly those dated April 1, 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.

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: 1. IP protocol and above, protocols that apply to the whole Internet. 2. 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. Comments and test results should be submitted 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 may 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 may, 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.

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 number (STD). 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 since all updates are made via RFCs and the RFC numbers are unique. Thus to clearly specify which version of a standard one is referring to, the standard number and all of the RFCs which 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 like "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, since 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 2800.

- **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 RFC1700.

- **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:
RFC1122 – Requirements for Internet hosts – communications layer,
RFC1123 – Requirements for Internet hosts – application and support, and
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.

1.3.2.1 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.

1.3.2.2 Obtaining RFCs

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

<http://www.ietf.org>

A complete list of current Internet Standards can be found in RFC 2800 – 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 Lima, Peru 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 via 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 would include the ruins of Macchu Picchu or offices in the Sistine Chapel. In these circumstances, fixed workstations would also benefit from otherwise unavailable network access.

Protocols such as Bluetooth, IEEE 802.11, and Wireless Application Protocol (WAP) are paving the way towards 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.

28 TCP/IP Tutorial and Technical Overview

Chapter 14. Wireless Application Protocol (WAP)

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 allows users of the referenced digital devices to securely access and interact with Internet, intranet, and extranet applications and services.

According to researchers, there will be more than 530 million wireless subscribers by the end of 2001. Some estimates report that the number of wireless subscribers will be more than one billion by the year 2004. Most of the phones sold in the near future will have multimedia capabilities that include:

- Retrieving e-mail
- Accessing data from company databases
- Executing stock trading
- Bill paying
- Making travel reservations
- Running other exciting applications

Based on these forecasts, wireless service providers, manufacturers of handheld devices, and software development companies see a huge new market of information-hungry users.

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 is contributing to various industry groups, like wireless service providers, handset manufacturers, infrastructure providers, and software developers. The forum has also provided the specifications to standards bodies like ARIB, CDG, ECMA, ETSI, TTA, and W3C. Today, more than 400 members, including IBM, belong to the WAP Forum.

14.1 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:

- Life of battery
- Less powerful CPUs
- Less memory
- Restricted power consumption
- Smaller displays
- Different input devices (for example, phone keypad and voice input)
- Lack of a mouse

Today's user application requirements will primarily be information inquiries rather than transaction processing. For example, users may only view what emails are 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, it is assumed that other tasks, like executing banking transactions, will be required in the not-so-distant future. This would require reliable data transmission, message privacy, integrity, and authentication. As a result, these functions must be included as part of the specifications of WAP.

14.2 Key elements of the WAP specifications

WAP defines an open standard architecture and set of protocols to implement wireless Internet access. The key elements of the WAP specification include:

- Definition of the WAP programming model (see Figure 191 on page 481)
- The Wireless Markup Language (WML) (see 14.3, "Wireless Markup Language (WML) and WMLScript" on page 486)
- A specification for a microbrowser
- A lightweight protocol stack (see 14.5, "Overview of the WAP protocol stack" on page 495)
- A framework for wireless telephony

14.2.1 Overview of the WAP programming model

The WAP architecture illustrated in Figure 191 describes:

- The WAP client (the handheld device or WAP terminal)
- The WAP gateway
- The Web server

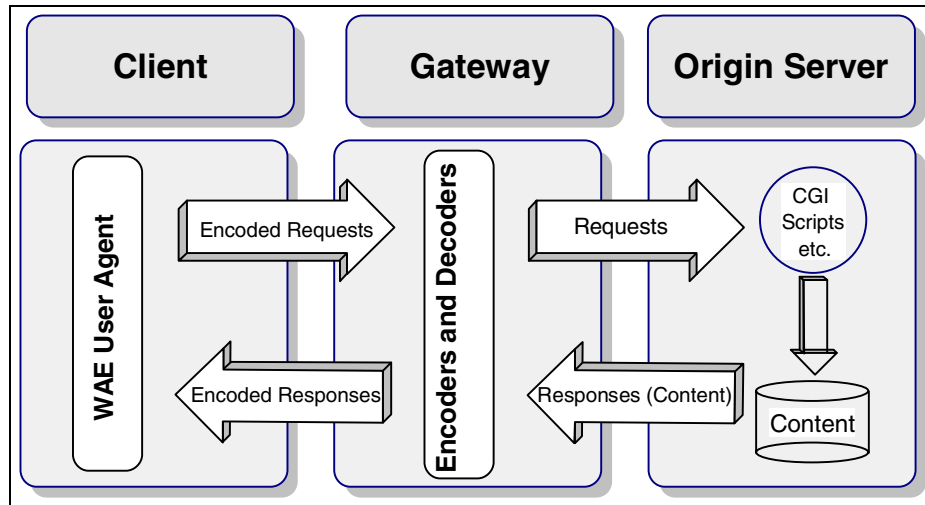


Figure 191. WAP programming model

This programming model is similar to the Web programming model. Programmers will see it as an advantage because they will meet a well-known interface based on client/server technology.

In order to solve the constraints of handheld devices (WAP terminals), there is a gateway protocol between the client and Web server. The gateway translates the traditional Web content into compact encoded formats. This reduces the size and number of packets traveling over the wireless network. Therefore, two connections are established via this gateway or proxy, with different protocols between client and proxy (WSP/WTP) and between proxy and Web server (TCP/IP HTTP).

14.2.1.1 WAP client

The client, also known as wireless application environment (WAE) user agent is a component of the WAP terminal. It consists of a microbrowser and the WAP protocol stack (see Figure 197 on page 496) in order to handle the execution of all requests and responses going through the WAP layered structure. For example this includes:

- Session establishment
- Data transport connection-less or connection-oriented
- Setting up a secure environment including:
 - Applying encryption and authentication
 - Encoding of outgoing requests
- Decoding of incoming responses to minimize bandwidth

The microbrowser is the interface to the user and to the WAP-content coming from the Internet or intranet. Two primary standards are supported by the microbrowser:

- Encoded Wireless Markup Language (WML)
- Compiled Wireless Markup Language Script (WMLScript)

See 14.3, “Wireless Markup Language (WML) and WMLScript” on page 486 for more details.

14.2.1.2 WAP gateway

The gateway works as a WAP proxy. It receives requests from the client, transforms an HTTP message, and sends it (based on the Uniform Resource Locator (URL)) to the addressed Web server. When the Web server returns the response, the gateway transforms it again into a bit-coded output. It then sends it to the mobile network, which directs it to the WAP client. This method allows the data content and applications to be hosted in standard Web servers using traditional Web technology.

The WAP gateway is also able to work as a WAP application server (see Figure 193 on page 486). This solution does not need a Web server. It can be used to support end-to-end security configurations. This could be for applications which require higher access control or a guarantee of service, like those used for Wireless Telephony Application (WTA) (see 14.5.2, “Wireless Telephony Application (WTA)” on page 498).

The WAP gateway decreases the response time to the WAP terminal by aggregating data from different Web servers and caching frequently used information. It also divides large HTTP responses into smaller transmission units before sending the responses to the client.

The WAP gateway can also interface with subscriber databases and use information to dynamically customize WML pages for a certain group of users.

The WAP gateway provides transition between the Internet and different non-voice mobile services. These might be services such as Short Message Services (SMS), Circuit Switched Data (CSD), or General Packet Radio Services (GPRS).

14.2.1.3 Origin server

The client's microbrowser requests Wireless Markup Language (WML) pages. These WML pages are stored on the origin server, which might be a Web server, connected via the Internet or intranet. WML pages may also be

stored in an application server installed in the gateway itself. A WML page consists of a WML deck. One WML deck is divided into one or more WML cards. A WML card can be regarded as a unit of interaction. Services let the user navigate back and forth between cards from one or several WML pages. WML, especially designed for WAP terminals, provides a smaller set of markup tags than HTML. It is based on HTTP 1.1. WML decks may also contain WMLScripts, another way of coding Web pages.

14.2.2 WAP network configurations

The WAP network may be designed using two configurations:

- The first configuration describes the client's access via a WAP gateway to a Web server. This configuration will be used when traditional Web services with HTTP connections are used.
- The second configuration allows the client to access a WAP application server directly without traversing an IP network. In this case the WAP gateway is not used. This solution will mainly be used for wireless telephony applications. These are applications, such as phone books, broadcast information about weather or traffic conditions, and stock quotes.

In order to save hardware costs, all functions of the WAP application server may also be implemented in the WAP gateway.

14.2.2.1 Usage of a WAP gateway

This type of configuration, illustrated in Figure 192, consists of two connections:

- The first connection is between the client and a WAP gateway.
- The second connection is between the WAP gateway and the Web server.

While the first connection uses the Wireless Session Protocol (WSP) and the Wireless Transaction Protocol (WTP), the second connection uses the traditional Hypertext Transfer Protocol (HTTP), which runs above TCP. The proxy gateway maintains the two connections as one logical connection between the client and the Web server.

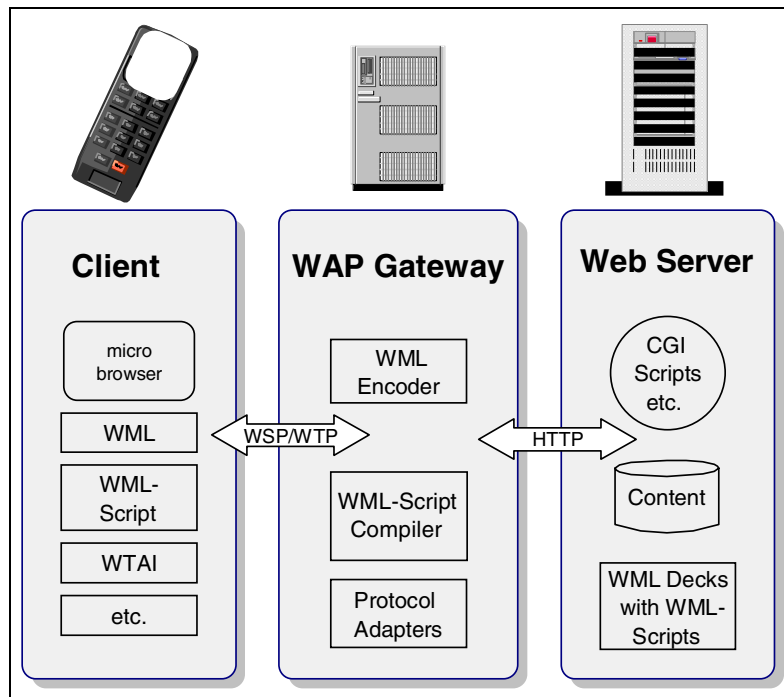


Figure 192. Usage of a WAP gateway

Client-server flow

The client-server flow is as follows:

1. A client asking for a particular service from an origin server submits a request to this server using the WML user agent (refer to Figure 198 on page 497).
2. The WML user agent uses the URL addressing scheme operation to find the origin server. URLs are used to address applications on a Web server, for example, a Common Gateway Interface (CGI).¹
3. The client's request is transmitted to the gateway using WSP/WTP.
4. The gateway does the bit-encoding, transforms the request into an HTTP message, and sends it to the Web server (addressed by the URL).
5. The origin server replies to the request by returning a single deck (refer to 14.3.1, "WML" on page 486), if it is stored in textual format in the origin server.
6. The deck is transmitted to the gateway.

¹ Essentially, the Common Gateway Interface (CGI) script is an executable entity that produces content as output.

7. On its way through the gateway, the textual format of each deck has to be converted into a format that is better suited for over-the-air transmission to WAP terminals. This conversion is done by the gateway using the WML encoder to convert the textual format into a binary format.
8. The gateway sends the encoded content to the client via the WAP protocol stack layers (refer to Figure 197 on page 496) over the wireless network.
9. At the client's WAP terminal, the data is received and can be displayed and interpreted, for example, by a microbrowser.

There are also some additional optimization procedures based on negotiation between the gateway and the client. The client may submit one or more additional requests via the WML user agent. These requests may also address WMLScripts, which can be received from the origin server at the WMLScript user agent.

The procedure in the gateway differs from the WML procedure. When the gateway receives a WMLScript, then:

- A WMLScript compiler takes the script as input
- Compiles it into a byte code
- Sends it to the WAP terminal, using the WAP protocol stack layer via the wireless network

14.2.2.2 Usage of a WAP application server

The usage of the WAP gateway is not mandatory. Encoding and compiling textual content need not be done in a gateway. It is conceivable that some origin servers have implemented WML encoders and WMLScript compilers. This could also be done in WAP application servers, as shown in Figure 193.

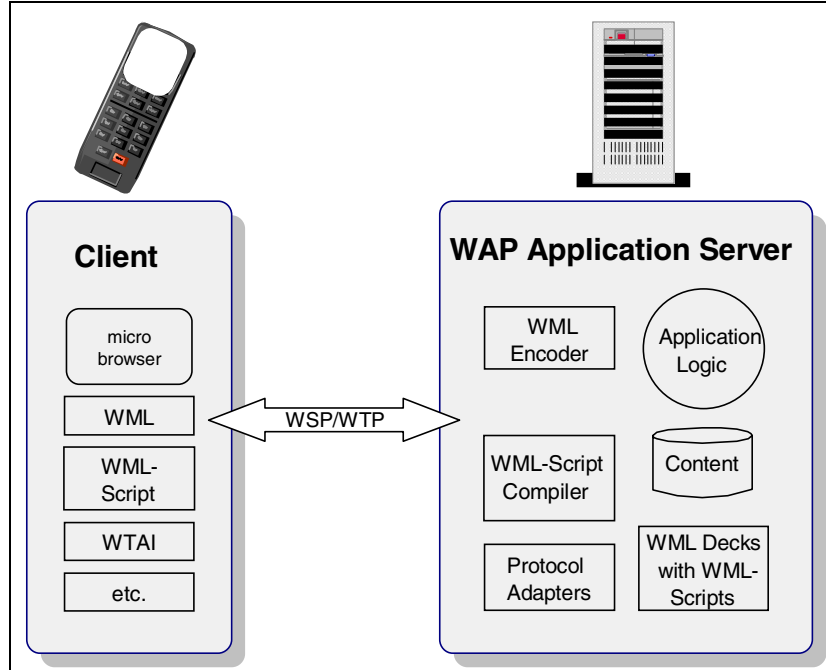


Figure 193. Usage of a WAP application server

14.3 Wireless Markup Language (WML) and WMLScript

Clients interact with origin servers through the WAP application entity (WAE) layer. WAE consist of user agents and services (see Figure 198 on page 497). The WML user agent offers two different services for communication between the client and the server:

- WML
- WMLScript

14.3.1 WML

WML is a tag-based document language. It is optimized for specifying presentation and user interaction on devices such as telephones and other wireless mobile terminals especially designed for narrow-band transmission. WML has implemented a deck and card structure. This allows the application to specify documents (commonly referred to as decks) made up of multiple cards. The user interacts with these WML cards in that it:

- Navigates forwards and backwards
- Reviews its contents

- May enter request information
- May meet choices
- Moves on to another card

There may be instructions which invoke services on origin servers within the cards. WML decks are stored as files on an origin server. They are fetched from the servers when required. The decks can also be dynamically generated by a content generator running on an origin server. Each card in a deck contains a specification for a particular user interaction.

WML provides a variety of features. For example:

- Support for text and images

It includes output presentation with emphasis elements (for example, bold, italic, big, and so on), with line breaks models (for example, line wrapping and line wrapping suppression), and tab columns that support simple tabbing alignment.

- Support for user input

Several elements to solicit user input are supported. The elements can be combined into one or more cards. WML includes text entry control for text and password input. Text entry fields may be masked, preventing the user from entering incorrect character types. WML provides options, such as selection control, that allows the user to present a list of options which can set data, navigate among cards, or invoke scripts. Also, task initiation controls may be set up.

- Navigation and history stack

WML allows several navigation mechanisms using URLs. Navigation includes HTML-style hyperlinks, intercard navigation elements, and history navigation elements.

- International support

WML uses the Universal Character set of ISO/IEC-1046. Currently, this character set is identical to Unicode 2.0.

- Narrow-band optimization

There are various technologies to optimize communication on a narrow-band terminal. For example, multiple user interactions (cards) are executed in one data transfer (a deck).

- State and context management

WML exposes a flat context control, which means that each WML text input can introduce variables. The state of the variables can be used to

modify the contents of a parameterized card without having to communicate with the server.

14.3.1.1 WML example

This example, illustrated in Figure 194, shows WML consisting of one deck, beginning with `<WML>` and ending with `</WML>`. The deck contains two cards. Cards begin with `<CARD>` and end with `</CARD>`.

The first card controls the welcome panel. The second card provides the input function including variables. It also gives the user the choice to select between two processing options.

The sample in Figure 194 navigates the user through the welcome panel. Two variables, *N* and *S*, are offered for the subsequent input. For the variable *S*, the user may enter 0 or 1.

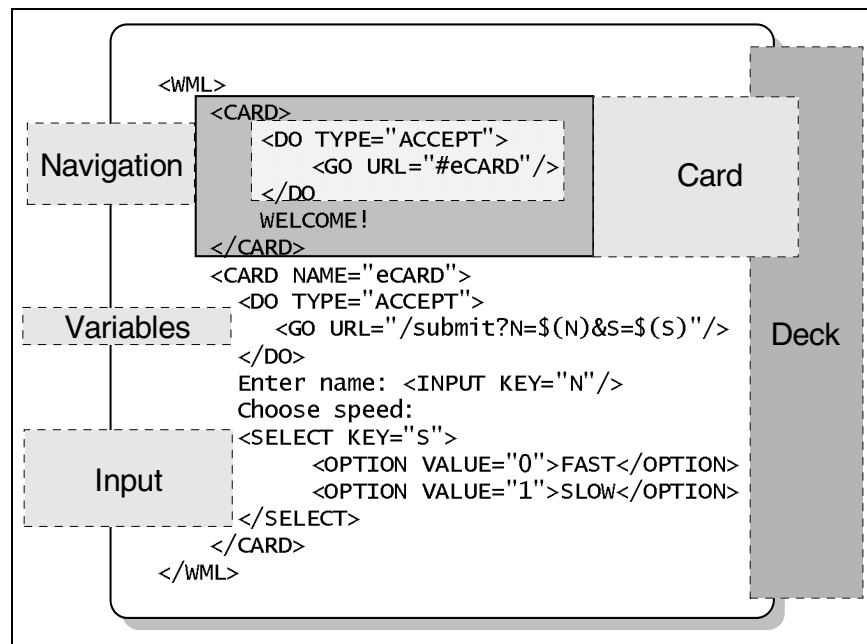


Figure 194. WML example

14.3.2 WMLScript

WMLScript is a lightweight procedural language. It enhances the standard browsing and presentation facilities of WML. For example:

- It adds intelligence to the client.

- It allows to create programming loops.
- It executes conditionals.

WMLScript is derived from JavaScript WWW scripting language. Thus, procedural logic can be brought to WML decks. WMLScript provides the application programmer with a variety of interesting capabilities:

- The ability to check the validity of user input before it is sent to the content server
- The ability to access device facilities and peripherals
- The ability to interact with the user without introducing round-trips to the origin server (for example, display an error message).

14.4 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*, as shown in Figure 195. 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.

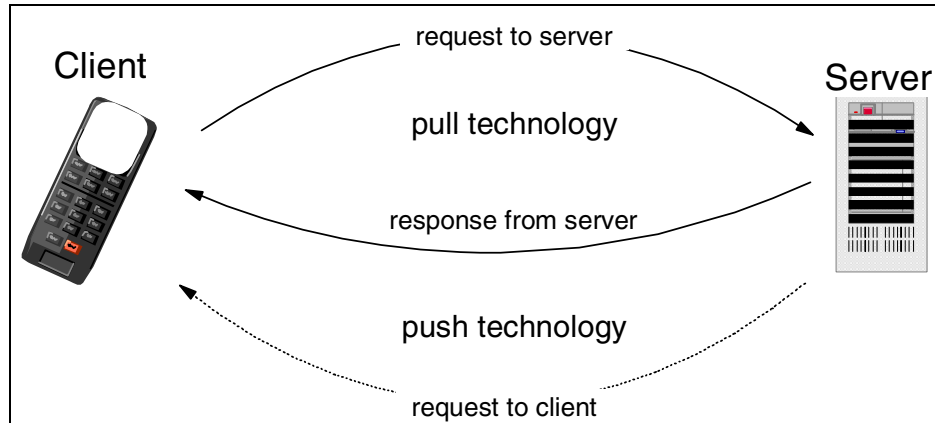


Figure 195. Comparison of pull vs. push technology

14.4.1 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 that the WAP client has subscribed to.

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 196.

Since 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).

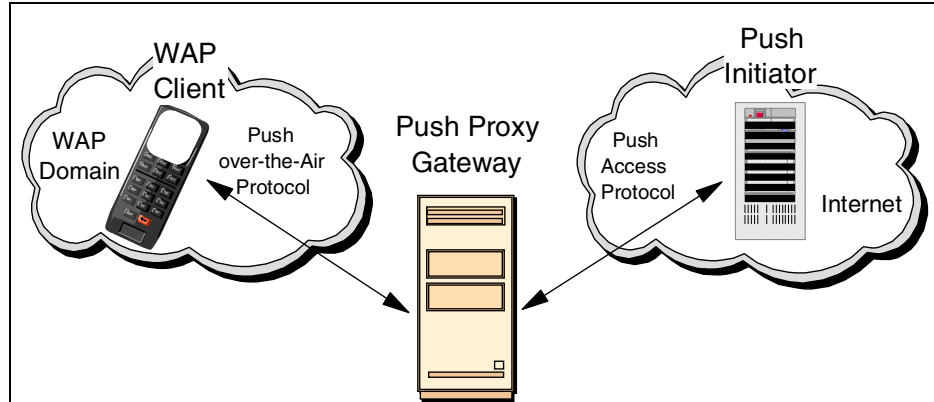


Figure 196. Push framework

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 may have the capability to tell the push initiator about the state of the delivery of the message to the client. But this may take some time, since the WAP client may not be online. If the client is online, it may accept or reject the pushed content.

The PAP uses XML messages, which may be tunneled through HTTP through the Internet.

14.4.2 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 may get access to the WAP domain and who is allowed to push messages.

14.4.2.1 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

14.4.2.2 Access from the Internet side

PPG accepts contents 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
- Time-outs
- Callback requests

The PPG will acknowledge the parsing of this control information. If the push initiator requests, it may 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).

14.4.2.3 PPG's 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 may be set by the push initiator or the policies of the mobile operator.

PPG may have implemented addressing aliasing schemes to enable special multi- and broadcast functions. This requires the translation of special addresses into broadcast operation. The push initiator may also query client capabilities and preferences to create a better formatted content for a particular WAP terminal.

14.4.3 Push access control protocol (PAP)

The push initiator uses PAP to push content via a PPG to the WAP domain. PAP is carried over an HTTP 1.1 tunnel. PAP carries an XML-style entity.

14.4.3.1 PAP operations

PAP supports the following operations:

- Push submission (initiator to PPG)

The push message contains a control entity, which provide 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 will convert this message into more bandwidth-optimized form before forwarding over-the-air through (OTA). Also, encrypting may be done 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 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 should 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.

14.4.4 Service indication

The service indication (SI) content type enables the sending of notifications to end users in an asynchronous manner (for example, new emails, and so on). An SI contains a short message and a uniform resource indicator (URI), indicating a service.

The message is presented to the end user upon reception. The end user may 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.

14.4.5 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 may 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, who listens to session requests from OTA-servers and responds by setting up a WSP session for the push purpose.

The client may verify the identity information of the OTA-server against a list of such servers before attempting any push session.

14.4.6 Client-side infrastructure

A connection-oriented push requires an active WSP session. As explained in 14.4.5, "Push over-the-air protocol (OTA)" on page 494, 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 may also ignore the session request from the PPG in case 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, in case the destination application is not installed. It is also responsible for confirming operations to the PPG.

14.4.7 Security

In a trusted environment, several questions may 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?

14.4.7.1 Authenticating a push initiator

Some of the following solutions may 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 push initiator and PPG, TSL or SSL can be used.

- Use of object-level certificates

Certificates could be used to sign and/or encrypt the pushed content on an end-to-end basis. This would strengthen 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, may be implemented.

- Combination of technologies

Another approach could be combining technologies by using TLS/SSL session with a PPG, while HTTP authentication could be used to authenticate the push initiator. Signed and/or encrypted content could then be sent over this authenticated session

14.4.7.2 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.

14.5 Overview of the WAP protocol stack

The WAP protocol stack consists of a number of protocols that provide functionality throughout the traditional networking layered stack model. This is outlined in Figure 197.

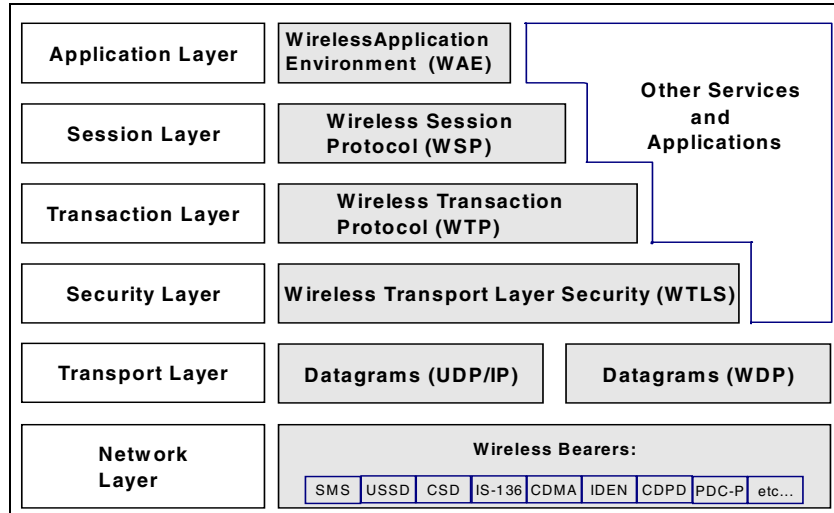


Figure 197. WAP protocol stack

14.5.1 Wireless application environment (WAE)

This layer contains similar WWW functions and mobile phone technologies. It includes a microbrowser environment that contains functionality such as WML, WML-Script, and Wireless Telephony Application (WTA).

WTA enables the user to interact with mobile-phone features and other user agents not specified by WAE, such as calender or phone book functions. For further information, see 14.5.2, “Wireless Telephony Application (WTA)” on page 498

WAE includes different user agents to execute different services (see Figure 198 on page 497). The user agent is the interface to the user and to the current selected service on the wireless device. WML, WMLScript, and WTA are examples of these services. The user agent uses a Uniform Resource Locator (URL) to address servers for a specific service content. The user agent sends a request with a URL for a service to the WAP gateway/proxy. The WAP gateway/proxy resolves the URL by an IP host address.

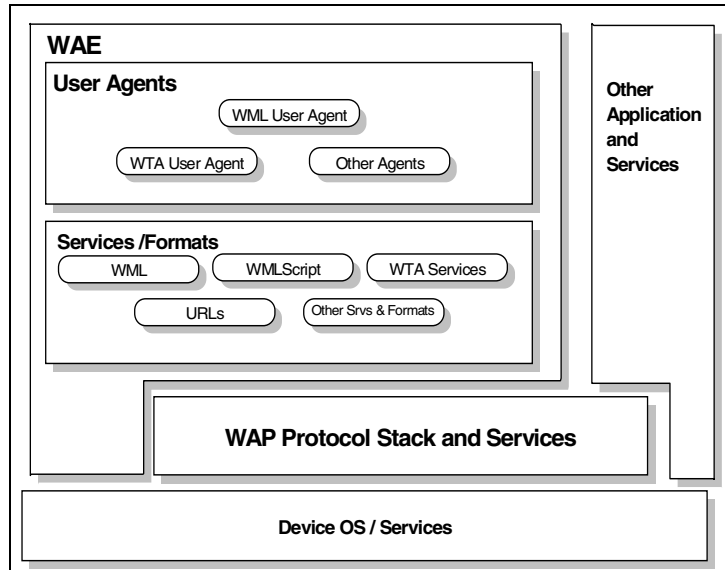


Figure 198. WAP client components

WAE defines a set of content formats for interoperable data exchange. The way of data exchange is dependent on the data and the targeted WAE 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, and availability of common tools, and so on.
- Multipart messages

It supports multipart-encoding scheme optimized for exchanging multiple typed content over WSP (see 14.5.3, “Wireless Session Protocol (WSP)” on page 498).
- User agent-specific formats

This is a function which supports two additional formats for exchanging data among user agents: electronic business card (vCard 2.1), and electronic calendaring and scheduling exchange format (vCalendar 1.0) specified by IMC.

WAE also defines WTA-specific formats to support wireless telephone applications.

14.5.2 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 are added to the standard WML/WMLScript browser to support an additional WTA Application Programming Interface (WTAI).

WTAI includes the following functions: call control, network text messaging, phone book interface, indicator control, and event processing.

Since WTA is beyond the TCP/IP scope, we will not discuss it further. Please see the official documentation of the WAP forum at:

www.wapforum.org

14.5.3 Wireless Session Protocol (WSP)

WSP establishes a reliable session between the client (see Figure 197 on page 496) and the server, which is the WAP gateway/proxy and releases that session in an orderly manner. WSP session establishment agrees on common level of protocol functionality using the capability of negotiation. WSP exchanges content between client and server using compact encoding, and also controls communication interrupt. Communication interrupt happens with change of a bearer, such as SMS (see Figure 197 on page 496).

WSP defines two protocols:

1. Connection-mode session services over a transaction service

This mode will be used for long-lived connections. A session state is maintained. There is reliability for data sent over a connection-mode session.

2. 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.

14.5.3.1 Basic functionality

The core of the WSP design is a binary form of HTTP 1.1.

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 overhead. 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 will pass 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 overhead of initial establishment. This allows a session to be suspended while idle to release network resources or save battery power. The session may be resumed over a different bearer network.

14.5.3.2 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.

For the service.type, see the following table:

Table 21. Service primitives

Type	Abbreviation	Description
Request	req	Used when a higher level requests a service from a lower level.

Type	Abbreviation	Description
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.

When using service primitives, additional parameters may be 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 may be negotiated between the client and the server.

- Push identifier

It 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 (for example, protocol error, disconnect mode, session suspended, and session resumed).

Service primitive sample

The behavior of service primitives is illustrated in Figure 199, using a time sequence chart.

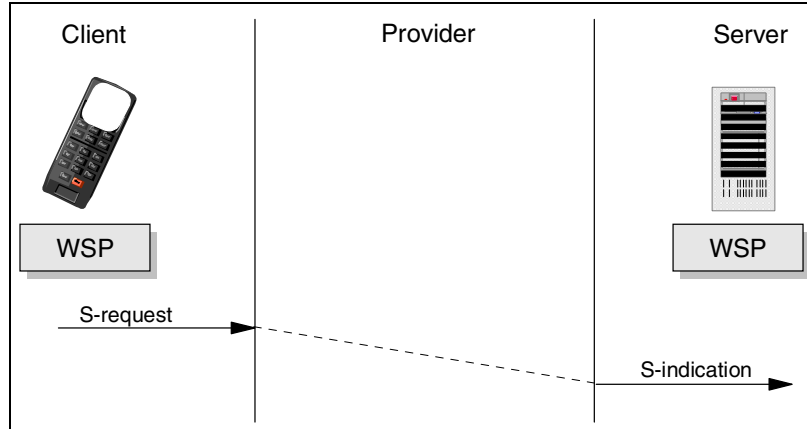


Figure 199. Non-confirmed service

Figure 199 depicts a simple non-confirmed service. The client invokes a 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.

14.5.3.3 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 may communicate with WDP (see 14.5.6, “Wireless Datagram Protocol (WDP)” on page 519) or with WTP (see 14.5.4, “Wireless Transaction Protocol (WTP)” on page 511), depending on the architecture of the application. Figure 200 on page 502 shows two sessions from a WAP client to a WAP proxy (further connections to Web server are not depicted).

One WAE application (see 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 (see solid line) uses additionally services from the transaction layer (WTP). This is because an additional class of services is required for particular transactions (see “WSP connection-mode using wireless transaction protocol (WTP)” on page 507).

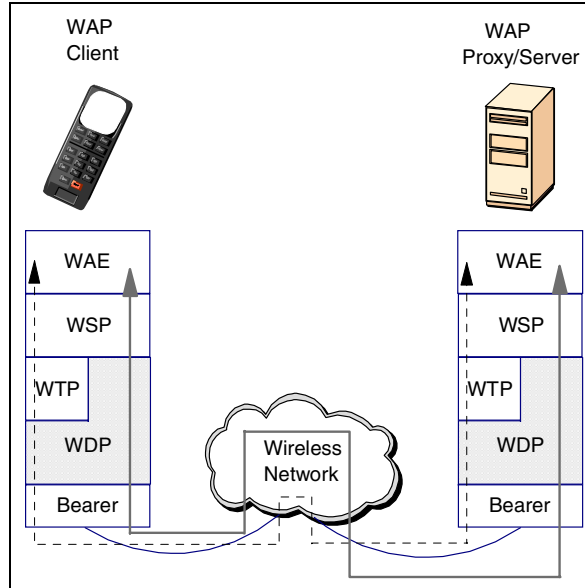


Figure 200. Client/server connection flow

WSP connection-mode - overview

Connection-mode session services provide two standard facilities:

- Session management facility

Session management facility allows a client to connect with a server based on agreement of facilities and options to use. During session establishment, the client may also exchange attribute information for the duration of the session. While 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

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

Method invocation facility permits the client to ask the server to execute an operation and return the result to the client. These operations may be compared with HTTP methods like GET, PUT, and so on (see RFC 2616), or user-defined operations, which fit into the same request/reply or

transaction pattern. The service users are notified about completion of the operation whether it was successful or failed.

- Push facility

Push facility allows the server to send unsolicited information to the client. The sent information is non-confirmed. Delivery may be not reliable.

- Confirmed push facility

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

Session resume facility allows both peers to suspend the session. The current session state is preserved. Further communication via 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 201 on page 505 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.

- Service primitives

- **S-Connect**

It is 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 It is 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* (tells which operation has to be used (either an HTTP method like 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* (contains 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 may 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 2613), the state of the requested operation invoked by S-MethodInvoke through the client), *response headers* (as described under S-MethodInvoke, the response body, the data associated with the response or, if the status indicates an error, contains further detailed error information), and *acknowledgment headers* (may be used to return some information back to the server).

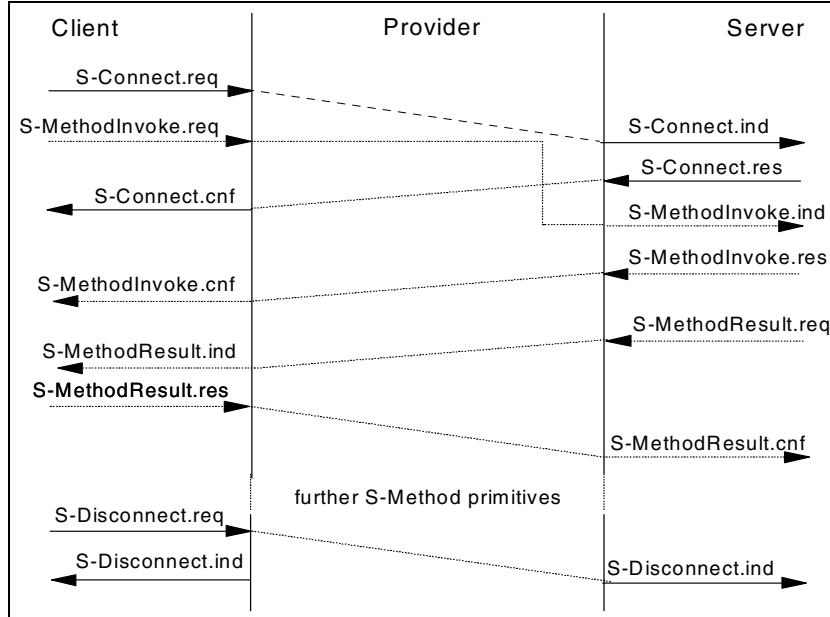


Figure 201. Normal session flow: establishment, actions and disconnection

- Usage of primitives:

Primitives are used to send an information (commands or data) to the session partner via the provider's network.

- In order to send an information, the primitive request type *****.req** is used.
- When the communication partner receives the sent information, it notifies the services in the same peer layer that a command is received and notifies the upper layer that there is data for the upper layer. This notification is done through a primitive of the type indication with a *****.ind** extension.
- If, in the session establishment process, the option *confirmation of requests* is negotiated, a primitive of the type response with the extension *****.res** is returned to the sender of the previous request.
- The receiver of the response informs its upper layer and its services in the same layer about the reception through a primitive of the type with a *****.cnf** extension.

The meaning of the extensions are:

- *****.req** = request

- `***.ind` = indication
 - `***.res` = response
 - `***.cnf` = confirm
- Description of the flow for a normal session establishment, session actions and session termination.

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.

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.

During the session initiation process, the client's session layer may 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.

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`.

After execution of the operation by the server, 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.

Since the session partner have negotiated confirmation of all requests (`***.req`), the client confirms the `S-MethodResult.req` request with an `S-MethodResult.res` response.

Further requests and responses may use this session.

The active session may be terminated in this example by issuing a `S-Disconnect.req` request to the server through the client. The server sends a `S-Disconnect.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-Disconnect.ind`.

Since the session layer does not provide any sequencing between multiple overlapping method invocations, the indications may 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 which are not used in the samples of the WSP connection-mode. These are:

S-Suspend	Suspends a session.
S-Resume	Resumes a session which 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-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

WSP connection-mode using wireless transaction protocol (WTP)

The following section describes the operation of when WSP sessions are run over wireless transaction protocol (WTP) services (see 14.5.4, “Wireless Transaction Protocol (WTP)” on page 511).

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 re-assembly of large messages and controlled data flow between sender and receiver.

Another advantage is the usage of WTP transaction classes. Three transaction classes allow you to define datagram transmission. These are:

Class 0 Unconfirmed invoke message with no result message, mainly used for unreliable push within a session using the same socket association.

- Class 1** Confirmed invoke message with no result message, which is mainly used by WSP to realize a reliable push service
- Class 2** Confirmed invoke message with one confirmed result message is the basic request/response transaction service used by WSP for method invocations.

The following table shows when these three transaction classes are used:

Table 22. 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:

- Normal session establishment using transaction class 2
- Normal session termination using transaction class 0
- Normal session suspend and resume using transaction class 0
- Normal method invocation using transaction class 2

Normal session establishment

Figure 202 shows the normal session establishment process.

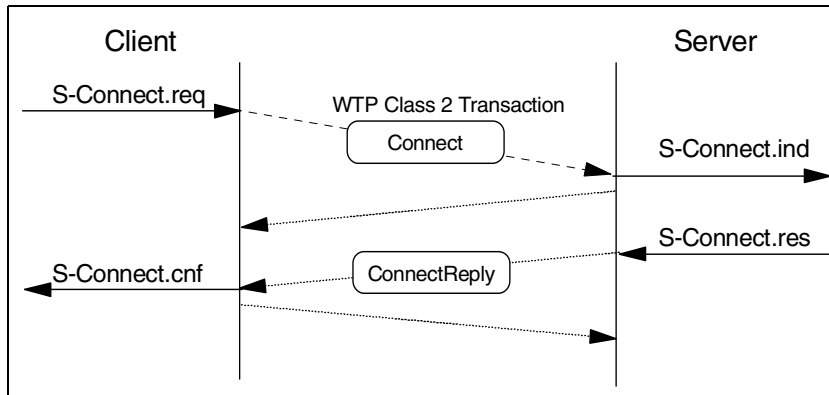


Figure 202. WSP WTP normal session establishment

Normal session termination

Figure 203 shows the normal session termination process.

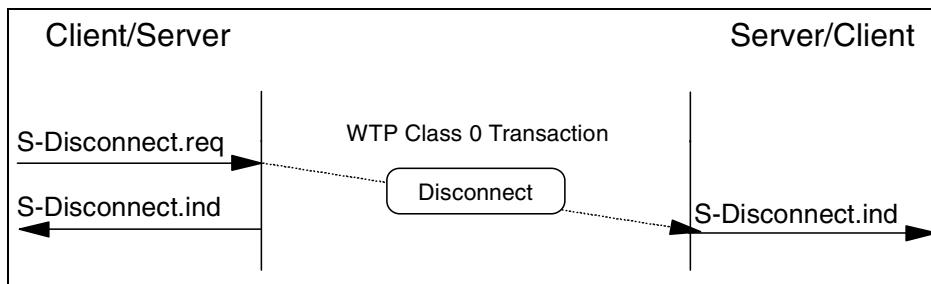


Figure 203. WSP WTP normal session termination

Normal session suspend and resume

Figure 204 shows the normal session suspend and resume process.

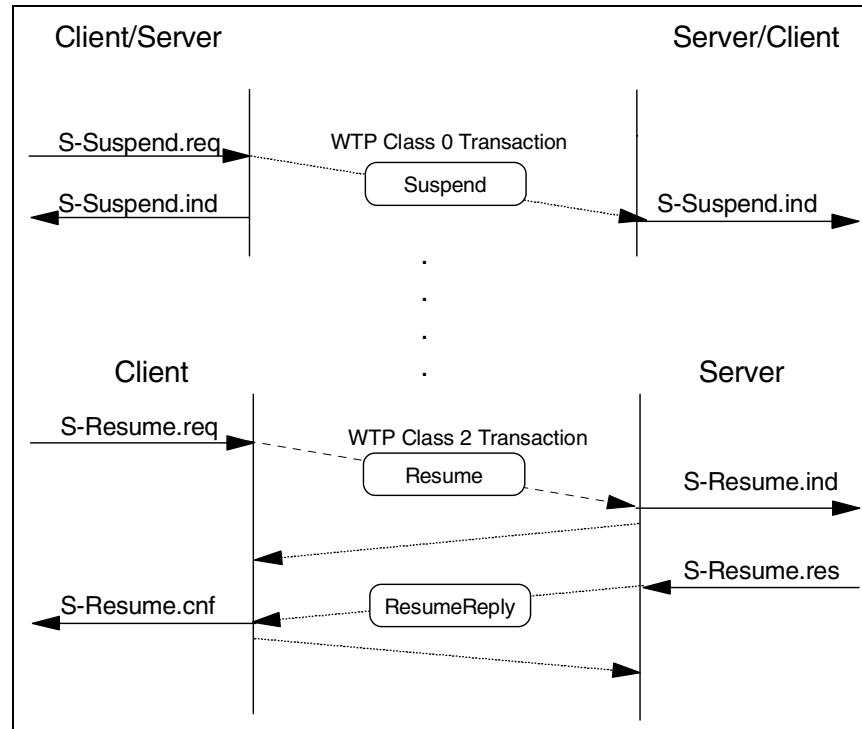


Figure 204. WSP WPT normal session suspend and resume

Normal method invocation

Figure 205 shows the normal method invocation process.

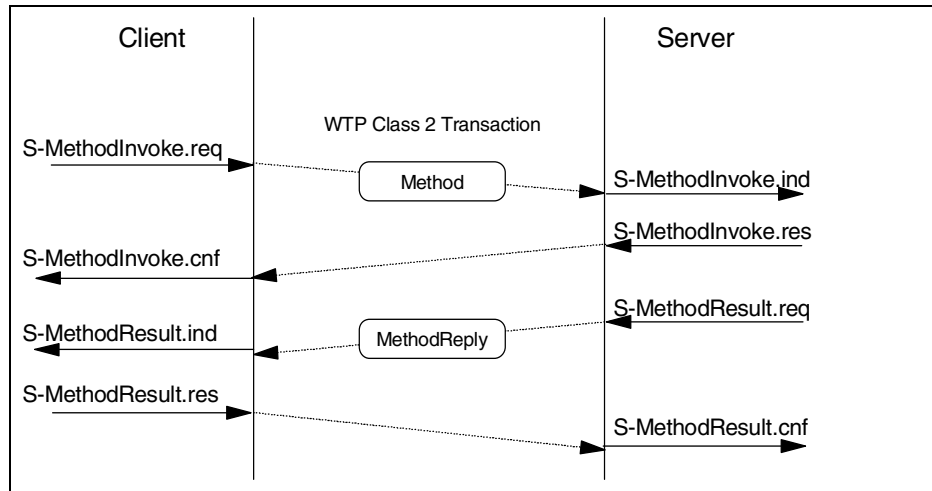


Figure 205. WSP WTP normal method invocation

14.5.3.4 Event processing

Sessions are distinguished by a 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 transaction 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.

14.5.4 Wireless Transaction Protocol (WTP)

WTP 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 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 are transmitted, WTP carries data already in the first packet of the protocol exchange.

WTP works message oriented, not stream oriented, like TCP.

14.5.4.1 WTP classes of operation

There are three classes of operation:

- Class 0: Unconfirmed invoke message with no result message
This is a Datagram that can be sent within the context of an existing WSP session.
- Class 1: Confirmed invoke message with no result message
This is used for data push, where no response from the destination is expected.
- Class 2: Confirmed invoke message with one confirmed result message
One single request produces a single reply

14.5.4.2 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 under WSP (see Table 21 on page 499).

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.

14.5.4.3 Example of a WSP-WTP sequence flow

Figure 206 on page 513 depicts the flow of a primitive sequence and shows the relationship between WSP and WTP request and responses. The flow is based on a class 2 transaction, a reliable, confirmed message exchange.

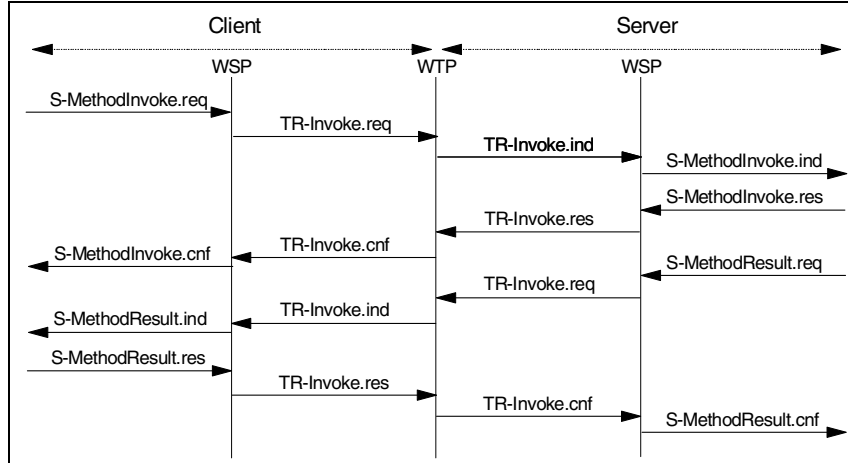


Figure 206. WSP-WTP primitive sequence for request-response

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.

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.

The third sequence returns the result request (reply to the inquiry) from the server to the client.

The fourth sequence confirms the receipt to the server that the reply was received correctly.

In order to control the message flow of the transaction layer peers WTP, uses two types of messages:

- Data messages that carry user data
- Control messages:
 - They are used for acknowledgments, error reporting, and so on.
 - Other control messages, which do not carry user data.

14.5.4.4 Summary of the WTP support

WTP supports:

- Retransmission of lost packets

- Selective retransmission
- Segmentation and re-assembly
- Port number addressing (UDP port numbers are used)
- Data flow control

14.5.5 Wireless Transport Layer Security (WTLS)

14.5.5.1 Overview

Many applications on the Web today require a secure connection between a client and the application server. WTLS is the security protocol to ensure secure transactions on WAP terminal. WTLS is based upon the industry-standard Transport Layer Security (TLS) protocol, formerly known as Secure Socket Layer (SSL).

WTLS 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 overhead. It provides end-to-end security and application-level security. This includes security facilities for en-/decrypting, strong authentication, integrity, and key management. WTLS complies with regulations on 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 207 on page 515 shows the location of WTLS in the WAP architecture model and the internal WTLS architecture with its different WTLS protocols.

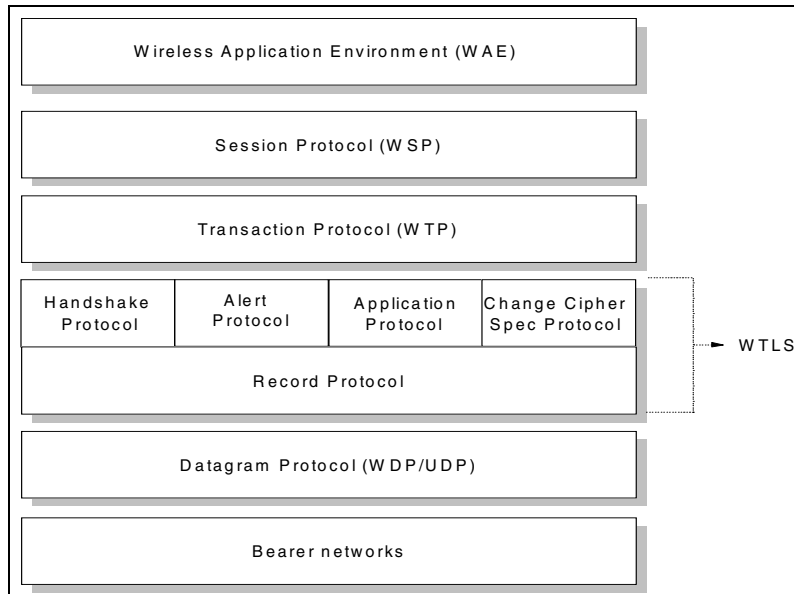


Figure 207. WTLS within the WAP architecture and WTLS components

14.5.5.2 WTLS layer goals

The primary goal is to provide security between 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 should be 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.

14.5.5.3 Overview of the WTLS protocols

WTLS operates on connection-oriented and/or datagram transport protocols. Security functions are executed as an option above the transport layer.

WTLS connection management

WTLS connection management provides a secure communication between 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 nearly similar to a secure connection establishment process via secure socket layer (SSL) (please see 21.8, “Secure Sockets Layer (SSL)” on page 747).

Security parameters for the secure connection establishment process may 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.
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

Figure 208 shows the location of WTLS within the WAP layer model and the WTLS protocols.

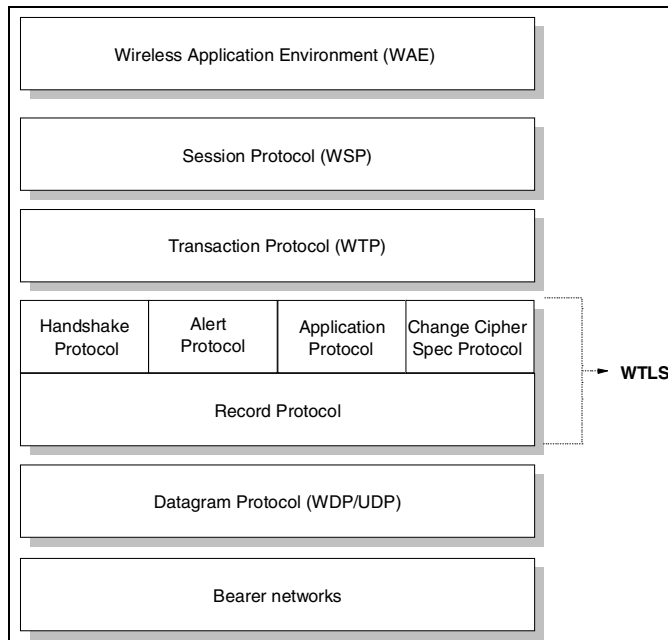


Figure 208. WTLS within the WAP architecture model and WTLS protocols

- Record protocol

The record protocol is the interface to the upper layer (transaction or session layer) and to the lower layer (transport layer).

The record protocol receives messages from the upper layer to be transmitted. It optionally compresses the data, applies a message authentication code (MAC), and encrypts and transmits the message.

Received data is decrypted, verified, decompressed, and delivered to a higher layer of the client.

Four protocols are defined that cooperate very closely with the record protocol in a client implementation environment:

- Handshake protocol

This protocol consists of three sub-protocols that allow peers to agree to security parameters for the record layer. The handshake protocol is responsible for the negotiation process between the client and server. 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.
Cipher spec	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 bytes 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.

- Change cipher spec protocol

The change cipher spec is sent by the client or server to notify the other partner that subsequent records will be sent under the newly negotiated cipher spec and keys. This message is sent during the handshake after the security parameters have been agreed upon, but before the verifying finished message is sent.

- Alert protocol

Alert messages contain information about the severity of the message and a description of the alert. Messages of a fatal level may terminate the secure connection.

- User data protocol

Consist of the payload which the WAP client intends to send.

Further information about cryptographic computations may be obtained from 21.2, “A short introduction to cryptography” on page 660.

14.5.6 Wireless Datagram Protocol (WDP)

WDP provides a consistent service to the upper layers (security, transaction, and session) of the WAP architecture. This allows for 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 may 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. Thus, WDP, with its kind of adaptation layer, cooperates with its underlying bearer layer. Figure 209 shows the general WDP structure.

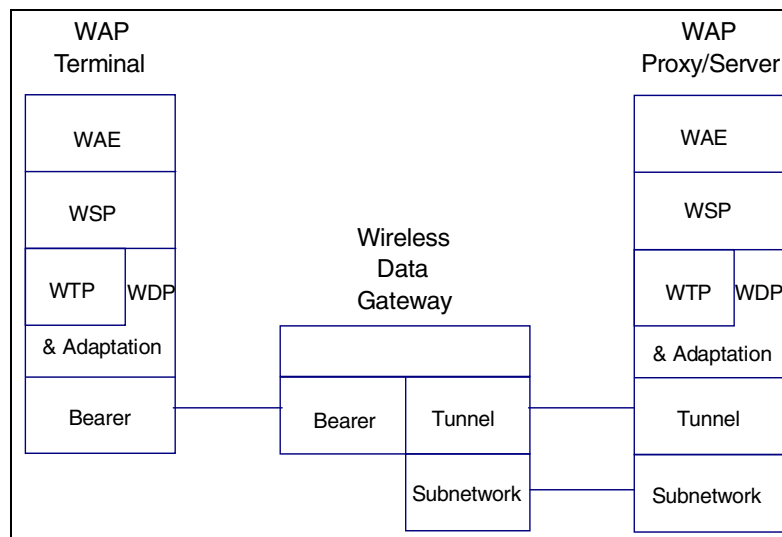


Figure 209. General WDP architecture

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 via 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 were 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 could be, for example, via a wide-area network

based on TCP/IP or frame relay, or a LAN operating TCP/IP over Ethernet. The WAP proxy/server may offer application content or may 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.

14.5.6.1 Bearer layer

The bearer layer is a bearer service, such as:

- IS-136 (GUTS, R-Data, DSD, and Packet Data)
- GSM (SMS, USSD, GPRS, and CSD)
- CDPD
- IDEN
- Flex and ReFLEX
- PHS
- PDC
- CDMA

14.5.6.2 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 the same as described for WSP (see Table 21 on page 499).

There are two primitives for the service layer:

T-DUnitdata Transmits data as datagram.

T-DError WDP may also receive a T-DError primitive if the requested transmission cannot be executed by the WDP protocol layer.

14.5.6.3 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 re-assembly.

The following bearer services are known to have adopted UDP as a WDP definition:

GSM Circuit-Switched Data, GSM GPRS, ANSI-136 R-Data, ANSI-136 Circuit-Switched Data, GPRS-136, CDPD, CDMA Circuit-Switched Data, CDMA Packet Data, PDC Circuit-Switched Data, PDC Packet Data, iDEN Circuit-Switched Data, iDEN Packet Data, PHS Circuit-Switched Data, DECT connection oriented services, and DECT packet switched services.

14.5.6.4 Mapping of WDP for other bearer services

There are a variety of services which map WDP, such as GSM SMS, ANSI-136 GHOST and USSD, and so on.

14.6 Protocol summary

Figure 210 shows the protocol structure of a WAP environment connected to a TCP/IP network. The purpose of this figure is to point out the different protocols used in the different environments, and what protocol conversion the WAP proxy/gateway has to do. In addition to this protocol conversion, the gateway has, after converting HTML code into WML code, to encode the content to a binary form in order to minimize bandwidth.

Protocol conversion is required only if HTML code is used for requests and responses on a connection to a Web server. If WML content or WMLScripts are retrieved from the Web server, then there is no protocol conversion necessary.

The WAP client initiates a session with the WAP proxy/gateway, using a reliable transaction, for retrieval of information from the Web server. The client selects, from the microbrowser's menu, the desired service by clicking on this line. This starts the transaction using an associated Uniform Resource Locator (URL), which identifies the Web target (IP address and port number for the Web server).

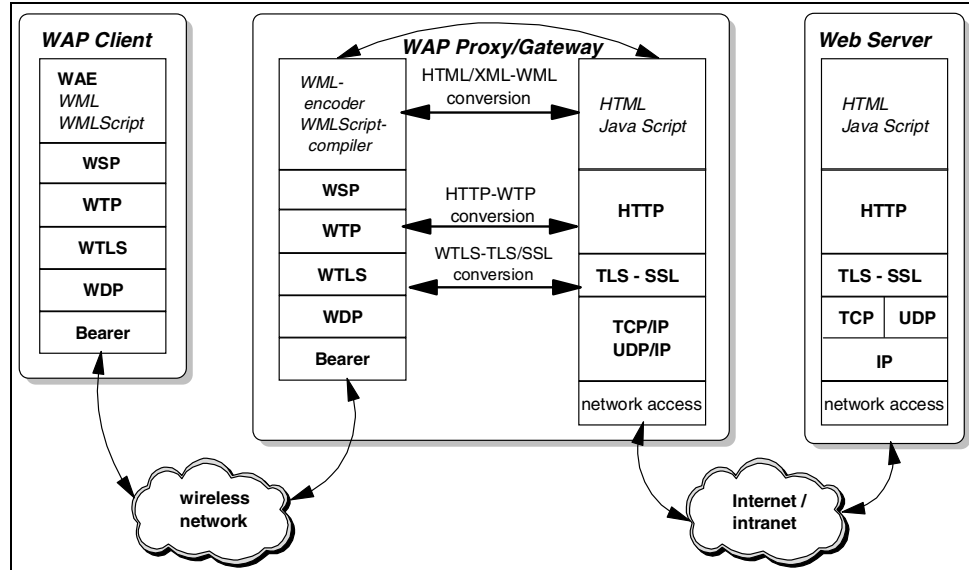


Figure 210. Protocol structure for a WAP and an Internet/intranet environment

If security is a concern for this session, a secure WTLS connection between the WAP client and the WAP proxy/gateway will be set up. Encryption, authentication and certificates are negotiated during the handshake process for the secure session. Please note that there is no secure end-to-end connection between the WAP client and the Web server, because each environment (the WAP and the TCP/IP environment) has different security protocols. If there is a security requirement for the connection between the WAP proxy/gateway and the Web server, a second secure connection path has to be established using TLS or SSL.

The WAP proxy/gateway will resolve the URL into a real IP address via internal cache or using domain name services, and will create the HTTP-, TCP-, and IP-header structure.

When the creation process for all headers is finished, the WAP proxy/gateway will set up an HTTP/TCP connection with the Web server and sends out a traditional IP datagram to the Internet/intranet. The path through the Internet or intranet is router controlled.

The Web server receives the HTTP request. It retrieves the desired Web page, WML deck, or WMLScript, and then responds to the HTTP request with its content to the WAP proxy/gateway, which is the other end of the HTTP/TCP connection.

The WAP proxy/gateway encodes the content (if it is WLM based), compiles the content (if it is WMLScript), or does protocol conversion into WML (if it is HTML code). It then associates the encoded response to the appropriate session and transaction, prepares the WSP/WTP/WDP header structure, and sends the data out to WAP client.

