

Uyless Black

TCP/IP and Related Protocols

- ICMP
- OSPF
- SNMP
- FTP
- RIP
- CMOT
- TELNET
- EGP
- Domain
- SMTP
- BOOTP
- Name
- ARP
- UDP
- System
- RARP
- IGP

TCP/IP

and Related Protocols

Black



G. SURESH

TCP/IP and Related Protocols

Uyless Black Series on Computer Communications

BLACK • *The V Series Recommendations: Protocols for Data Communications Over the Telephone Network*

BLACK • *The X Series Recommendations: Protocols for Data Communications Networks*

BLACK • *Network Management Standards: The OSI, SNMP and CMOL Protocols*

HUGHES • *Data Communications*

KIMBERLY • *Electronic Data Interchange*

RADICATI • *Electronic Mail: An Introduction to the X.400 Message Handling Standards*

WHITE • *Internetworking and Addressing*

TCP/IP and Related Protocols

Uyless Black

McGraw-Hill, Inc.

**New York St. Louis San Francisco Auckland Bogotá
Caracas Lisbon London Madrid Mexico Milan
Montreal New Delhi Paris San Juan São Paulo
Singapore Sydney Tokyo Toronto**

Library of Congress Cataloging-in-Publication Data

Black, Uyless D.

TCP/IP and related protocols / Uyless Black.

p. cm. — (Uyless Black series on computer communications)

Includes index.

ISBN 0-07-005553-X

1. Computer networks. 2. Computer network protocols. I. Title.

II. Series.

TK5105.5.B5663 1992

004.6'2—dc20

91-37187

CIP

*McGraw-Hill Series on Computer Communications, Uyless Black,
Series Advisor*

Copyright © 1992 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

4 5 6 7 8 9 0 DOC/DOC 9 8 7 6 5 4 3

ISBN 0-07-005553-X

The sponsoring editor for this book was Neil Levine, the editing supervisor was Fred Dahl, and the production supervisor was Pamela A. Pelton. It was set in Century Schoolbook by McGraw-Hill's Professional Book Group composition unit.

Printed and bound by R. R. Donnelley & Sons Company.

Information contained in this work has been obtained by McGraw-Hill, Inc., from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

To my wife Holly, for all her efforts in the building of our new home (and just about everything else). She kept the contractors going while I finished this book and completed a long lecture tour. I could not possibly do all that I do without her.

Contents

Preface	xv
Acknowledgments	xvii

Chapter 1. TCP/IP and the Internet	1
Introduction	1
Organization of Internet	3
Request for Comments (RFCs)	4
TCP/IP and the OSI	4
Internetworking Architecture	5
Terms and Concepts	5
Connectionless and Connection-Oriented Protocols	6
The Internet Layers	9
Example of the Layer Operations	10
The TCP/IP Model: A Closer Look	12
Ports and Sockets: An Introduction	12
Challenges of Internetworking	13
Typical Internet Topologies	15
Summary	18
 Chapter 2. Introduction to Networks, Bridges, Gateways, and Routers	 19
Introduction	19
A General Taxonomy	19
Wide Area Networks (WANs)	22
Circuit Switching	22
Message Switching	22
Packet Switching	23
Local Area Networks (LANs)	27
LAN Components	27
Types of LANs	28
Carrier Sense, Collision Detection	28
CSMA/CD and IEEE 802.3	29
Token Ring	31
Token Bus	33
The LLC Sublayer	33
Classes of Service	34
Repeaters, Bridges, Routers, Brouters, and Gateways	34

Source Routing and Spanning Tree Bridges	36
Summary	36
Chapter 3. Naming, Addressing, and Routing in an Internet	39
Introduction	39
Upper Layer, Network, Data Link, and Physical Names and Addresses	40
Physical Addresses	40
Universal Physical Addresses and Protocol Identifiers	41
The CSMA/CD Frame and MAC Physical Addresses	44
Link Layer Addresses (LSAPs)	45
Extension to the LSAP Header (SNAP)	46
Network Addresses	47
Physical and Network Address Resolution	48
Are All These Addresses Necessary?	49
Upper Layer Addresses and Names	50
Example of a Complete Naming and Addressing Operation	50
The IP Address Structure	54
Destination Addresses and Routing	57
Direct and Indirect Destinations	57
IP Routing Logic	57
Multihomed Hosts	60
Address Resolution Issues	60
The Address Resolution Protocol (ARP)	61
ARP Address Translation Table	65
Proxy ARP	66
Reverse Address Resolution Protocol (RARP)	67
Primary and Secondary RARPs	68
IP and X.121 Address Mapping	69
International Numbering Plan for Data Networks (X.121)	69
DDN IP Addresses	69
Mapping X.121 and IP Addresses	70
Subnets, Subnet Addressing, and Address Masking	70
Subnet Masks	73
Summary	75
Chapter 4. The Domain Name System (DNS)	77
Introduction	77
The Domain Name System (DNS) Architecture	78
Domain Name Space	79
Top-Level Domains	80
Domain Name Resolution and Mapping Names to Addresses	80
Name Server Operations	82
Resource Records (RR)	83
RDATA Field	85
Explanation of DNS Types	86
IN-ADDR-ARPA	88
Example of a Resource Record	90
DNS Messages	93
RR Compression	96
Summary	96

Chapter 5. The Internet Protocol (IP)	87
Introduction	97
Major Features of IP	97
IP and Subnetworks	98
The IP Datagram	99
Major IP Services	106
Internet Header Check Routine	106
IP Source Routing	107
Route Recording Option	109
The Timestamp Option	109
Fragmentation and Reassembly	111
IP Address and Routing Tables	116
Address Table	116
Routing Table	117
IP Service Definitions and Primitives	119
IP/ULP Primitives	119
IP/SNP Primitives	122
Network Layer/LLC Primitives	122
Multicasting	124
Internet Group Management Protocol (IGMP)	126
Other Thoughts on IP	127
Connectionless-Mode Network Service (ISO 8473)	127
The ISO 8473 PDU	128
Quality of Service (QOS) Functions	128
Protocol Functions	129
Traffic Management between Subnetworks	129
CLNP and IP	133
Summary	134
 Chapter 6. Internet Control Message Protocol (ICMP)	 135
Introduction	135
ICMP Message Format	137
ICMP Error- and Status-Reporting Procedures	137
Time Exceeded	138
Parameter Unintelligible	139
Destination Unreachable	139
Source Quench	140
Echo Request and Reply	140
Redirect	141
Timestamp and Timestamp Reply	142
Information Request and Information Reply	143
Address Mask Request and Address Mask Reply	143
Other Thoughts on ICMP	144
Summary	144
 Chapter 7. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)	 145
Introduction	145
Value of the Transport Layer	145

TCP Overview	146
Major Features of TCP	147
Another Look at Ports and Sockets	180
Examples of Port Assignments and Port Bindings	150
Passive and Active Opens	153
The Transmission Control Block (TCB)	154
TCP Window and Flow-Control Mechanisms	154
Retransmission Operations	156
Estimating Timers for Time-Outs and Retransmissions	160
TCP and User Interfaces	162
The Segment (TCP PDU)	162
The TCP Connection Management Operations	167
Examples of the TCP Open, Data Transfer, and Close Operations	167
TCP Connection Table	174
Other Considerations in Using TCP	176
User Datagram Protocol (UDP)	177
Format of the UDP Message	177
Summary	179
 Chapter 8. Route Discovery Protocols	 181
Introduction	181
Terms and Concepts	181
Routing Based on Fewest Hops	182
Routing Based on Type of Service Factors	184
Core and Noncore Gateways	186
Exterior and Interior Gateways	186
Border Routers and Boundary Routers	187
How Autonomous Systems or Areas Exchange Information	188
Who Participates in Exchanging Routing Information and in Executing	
Routing Logic?	190
Clarifying Terms	191
The Gateway-to-Gateway Protocol (GGP)	192
Introduction	192
Neighbor Connectivity Analysis	193
Exchanging Routing Information	193
Computing Routes	194
GGP Message Formats	196
Example of a GGP Update Message	199
External Gateway Protocol (EGP)	199
Introduction	199
Major Operations of EGP	201
The EGP States	201
Types of Messages	202
EGP Message Types	203
Example of an EGP Update Message	208
EGP Events	210
EGP Operations During the Up State	210
Other Features of EGP	213
Interior Gateway Protocols (IGPs)	214
Routed (Routing) Information Protocol (RIP)	214
RIP Timers	215
Example of RIP Operations	216

RIP Problems	216
The RIP Message Format	218
The RIP Request and Response	218
RIP versus OSPF	219
The Hello Protocol	219
The Hello Message Format	220
Gated	221
Summary of Vector-Distance Protocols	221
Choosing the Optimum Path with a Shortest Path Algorithm	221
Open Shortest Path First (OSPF) Routing Protocol	226
Introduction	226
OSPF Operations	227
Classification of Routers	229
Types of Advertisements	230
Example of the OSPF Shortest Path Tree	230
OSPF Data Structures	233
The OSPF Packets	237
Summary	249
Chapter 9. The Major Application Layer Protocols	251
Introduction	251
The TELNET Protocol	251
Network Virtual Terminal	252
TELNET RFCs	253
TELNET Commands	256
Example of TELNET Commands	257
Trivial File Transfer Protocol (TFTP)	258
TFTP and Other Protocols	258
TFTP Packets	259
File Transfer Protocol (FTP)	261
Data Types	263
FTP Commands and Replies	263
Sequence of Operations in an FTP Session	265
Examples of FTP Operations	267
Example of a File Retrieval	269
Minimum Implementation of FTP	269
Simple Mail Transfer Protocol (SMTP)	271
SMTP Model	271
Address Field Format	272
Examples of SMTP Operations	273
SMTP and the Domain Name System (DNS)	274
Summary	275
Chapter 10. Other Protocols	277
Introduction	277
X Windows	277
The X Window System Protocol	279
A Display Connection	280
Remote Procedure Call (RPC)	281
Network File System (NFS)	282
The NFS Server Procedures	282

Remote Exec Daemon (REXECD)	283
PING	283
HOSTNAME	284
Host Monitoring Protocol (HMP)	284
Discard Protocol	285
Finger	286
The Bootstrap Protocol (BOOTP)	286
Network Time Protocol (NTP)	288
Daytime	290
Summary	290
Chapter 11. Internet Network Management Systems	293
Introduction	293
Summary of the Internet Network Management Standards	293
Layer Architecture for Internet Network Management	294
The Internet Naming Hierarchy	295
Structure of Management Information (SMI)	297
SMI Syntax and Types	297
The Management Information Base (MIB)	298
Overview of the Object Groups	299
Templates to Describe Objects	302
Definition of High-Level MIB	303
The SNMP	304
SNMP Administrative Relationships	304
Example of an SNMP Operation	306
SNMP PDUs	306
The SNMP MIB Managed Objects	308
CMOT	308
The CMOT Layers	310
The Lightweight Presentation Protocol (LPP)	311
Summary	311
Chapter 12. Operating TCP/IP with Other Protocols (and Other Protocols without TCP/IP)	313
Introduction	313
A Minimum TCP/IP and LAN Stack	314
A Word about Operating System Dependency	314
TCP/IP over LLC	315
Replacing TCP with UDP	316
NetBIOS over TCP or UDP	317
IP over NetBIOS	318
XNS over IP	319
IP Router Stacks	320
Relationship of IP and LAN Bridges	321
IP and X.25	321
IP, X.25, and LANs	321
IP, X.25, and Public Data Networks (PDNs)	322
IP, X.25, and Amateur Packet Radio	323
Using IPX with UDP/IP Networks	323

Transmitting 802 LLC Traffic over IPX Networks	324
Transmitting IP Datagrams over ARCNET Networks	325
Transmitting IP Datagrams over FDDI Networks	326
IP over Switched Megabit Data Service (SMDS)	327
OSI's Transport Protocol Class 0 over TCP	328
OSI Connectionless Transport Layer over UDP	329
TCP/IP over ISDN	330
Stacks for the 1990s	331
Summary	332
Chapter 13. TCP/IP and Operating Systems	333
Introduction	333
UNIX and TCP/IP	333
Connection-Oriented Services	334
Other Input/Output Calls	337
Datagram Services	338
Closing a Connection	338
Other System Calls	339
Example of Programs to Invoke UNIX-based TCP/IP Services	339
PC Interface Program	339
Sending Mail Through SMTP	341
Sending a File Through TFTP	344
Sending a File Through FTP	345
Using tn to Invoke TELNET Login Services	346
Retrieving Network Statistics	347
Using PING to Obtain Echo Services	347
Using Route to Manipulate the IP Routing Table	348
Summary	349
Chapter 14. Management Considerations	351
The Vendor Strategies with the Internet-Based Products	351
The Vendors React to the Marketplace	352
A Comparison of the TCP/IP and OSI Stacks	353
IP and CLNP	356
TP4 and TCP	356
Migration Issues	358
Gateway Migration Issues	359
Network Migration Issues	359
Host Migration Issues	361
Migration of Internet to OSI Network Architectures	361
Internetworking DOD and Non-DOD Architectures	361
Dual DOD/OSI Protocol Hosts	362
Application Layer Gateways	362
Activities of the Vendors	362
Limited OSI Capability	363
Equivalent OSI Capability	363
Advanced OSI Capability	363
Summary	364

The Internet Protocol (IP)

Introduction

As we learned in Chap. 1, the Internet Protocol (IP) is an inter-networking protocol developed by the Department of Defense. The system was implemented as part of the DARPA internetwork protocol project and is widely used throughout the world. This chapter examines IP in more detail including its major features, its use of address and routing tables, and its relationships to other internet and IEEE protocols.

Major Features of IP

IP is quite similar to the ISO 8473 (the Connectionless Network Protocol or CLNP) specification explained in the last section of this chapter. Many of the ISO 8473 concepts were derived from IP.

IP is an example of a connectionless service. It permits the exchange of traffic between two host computers without any prior call setup. (However, these two computers usually share a common connection-oriented transport protocol.) Since IP is connectionless, it is possible that the datagrams could be lost between the two end users' stations. For example, the IP gateway enforces a maximum queue length size, and if this queue length is violated, the buffers will overflow. In this situation, the additional datagrams are discarded in the network. For this reason, a higher-level transport layer protocol (such as TCP) is essential to recover from these problems.

IP hides the underlying subnetwork from the end user. In this context, it creates a virtual network to that end user. This aspect of IP is attractive because it allows different types of networks to attach to an IP gateway. As a result, IP is reasonably simple to install, and because of its connectionless design, it is quite robust.

Since IP is an unreliable, best effort, datagram-type protocol, it has no reliability mechanisms. It provides no error recovery for the underlying subnetworks. It has no flow-control mechanisms. The user data (datagrams) may be lost, duplicated, or even arrive out of order. It is not the job of IP to deal with most of these problems. As we shall see later, most of the problems are passed to the next higher layer, TCP.

IP supports fragmentation operations. The term *fragmentation* refers to an operation wherein a protocol data unit (PDU) is divided or segmented into smaller units. This feature can be quite useful because all networks do not use the same size PDU. For example, X.25-based wide area networks (WANs) typically employ a PDU (called a packet in X.25) with a data field of 128 octets. Some networks allow negotiations to a smaller or larger PDU size. The Ethernet standard limits the size of a PDU to 1500 octets. Conversely, proNET-10 stipulates a PDU of 2000 octets.

Without the use of fragmentation, a gateway would be tasked with trying to resolve incompatible PDU sizes between networks. IP solves the problem by establishing the rules for fragmentation at the gateway(s) and reassembly at the receiving host.

IP and Subnetworks

As seen in Fig. 5.1, IP is designed to rest on top of the underlying subnetwork—insofar as possible in a transparent manner. This means that IP assumes little about the characteristics of the underlying network or networks. As stated earlier, from the design standpoint this is quite attractive to engineers because it keeps the subnetworks relatively independent of IP.

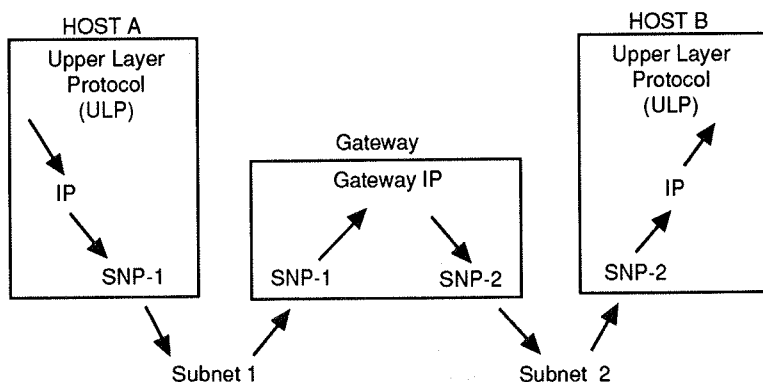


Figure 5.1 The IP Model.

As the reader might expect, the transparency is achieved by encapsulation. The data sent by the host computer are encapsulated into an IP datagram. The IP header identifies the address of the receiving host computer. The IP datagram and header are further encapsulated into the specific protocol of the transit network. For example, a transit network could be an X.25 network or an Ethernet LAN.

After the transit network has delivered the traffic to an IP gateway, its control information is stripped away. The gateway then uses the destination address in the datagram header to determine where to route the traffic. Typically, it then passes the datagram to a subnetwork by invoking a subnetwork access protocol (for example, Ethernet on a LAN, and X.25 on a WAN). This protocol is used to encapsulate the datagram header and user data into the headers and trailers that are used by the subnetwork. This process is repeated at each gateway, and eventually the datagram arrives at the final destination, where it is delivered to the receiving station.

It should be emphasized that an IP gateway is not completely unconcerned with the access protocol of an attached subnetwork. There is nothing magic about IP; the gateway interface must know how to access its connected networks. Consequently, some form of communications (however limited) must occur between the gateway and the attached subnetworks (see the section later in this chapter titled "IP Service Definitions and Primitives"). The important point is that an IP module does not care about the operations inside the networks.

The IP Datagram

A productive approach to the analysis of IP is first to examine the fields in the IP datagram (PDU) depicted in Fig. 5.2. An abbreviated description of the fields in the datagram is provided in Table 5.1.

The *version* field identifies the version of IP in use. Most protocols contain this field because some network nodes may not have the latest release available of the protocol. The current version of IP is 4.

The *header length* field contains 4 bits set to a value to indicate the length of the datagram header. The length is measured in 32-bit words. Typically, a header without QOS options contains 20 octets. Therefore, the value in the length field is usually 5.

The *type of service (TOS)* field can be used to identify several QOS functions provided for in Internet. It is quite similar to the service field that resides in the CLNP PDU. Transit delay, throughput, precedence, and reliability can be requested with this field.

The TOS field is illustrated in Fig. 5.3. It contains five entries consisting of 8 bits. Bits 0, 1, and 2 contain a precedence value used to

Version (4)	Header Length (4)
Type of Service (8)	
Total Length (16)	
Identifier (16)	
Flags (3)	Fragment Offset (13)
Time to Live (8)	
Protocol (8)	
Header Checksum (16)	
Source Address (32)	
Destination Address (32)	
Options and Padding (Variable)	
Data (Variable)	

(n) = Number of Bits in Field

Figure 5.2 The IP Datagram.

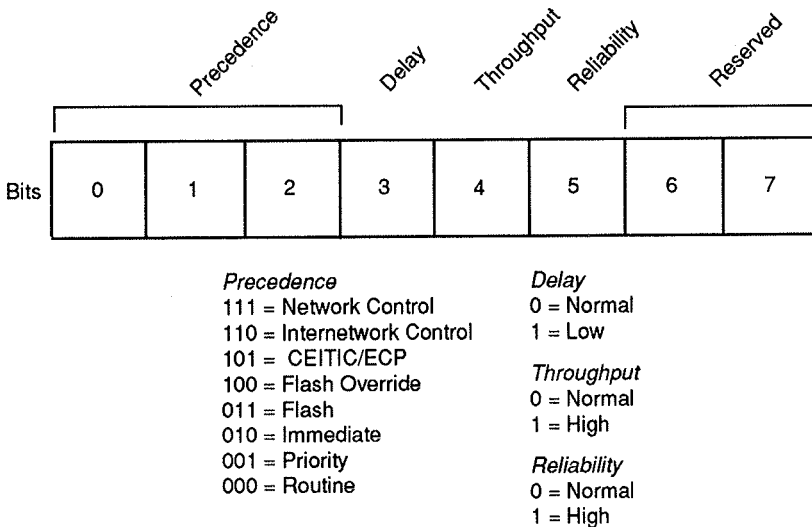
indicate the relative importance of the datagram. Values range from 0 to 7, with 0 set to indicate a *routine precedence*. The precedence field is not used in all systems, although the value of 7 is used by some implementations to indicate a network control datagram. However, the precedence field could be used to implement some powerful flow control and congestion mechanisms in a network. This would allow gateways and host nodes to make decisions about the order of “throwing away” datagrams in case of congestion.

The next three bits are used for other services and are described as follows:

Bit 3 is the *delay bit (D bit)*. When set to 1 this TOS requests a short delay through an internet. The aspect of delay is not defined in the standard, and it is up to the vendor to implement the service.

TABLE 5.1 The IP Protocol Data Unit

Version field	Identifies the version of IP.
Internet header length	Specifies the length of the IP header.
Type of service	Stipulates quality of service functions.
Total length field	Specifies the total length of the IP datagram, including the header.
Identifier	Used with the address fields to identify the data unit uniquely (for fragmentation).
Flags	Used in the fragmentation operations.
Fragmentation offset	Describes where this datagram belongs within the original PDU.
Time to live	Used to determine how long the datagram is to remain in the internet.
Protocol	Used to identify a next-level protocol that is to receive the user data at the final destination.
Header checksum	Used to perform an error check on the header.
Source and destination addresses	Identifies the source and destination hosts and their directly attached networks.
Options	Used to request additional services for the IP user.
Padding	Used to give the datagram a 32-bit alignment.
User data	Contains user data.


Figure 5.3 The Type of Service (TOS) Field.

The next bit is the *throughput bit* (*T bit*). It is set to 1 to request high throughput through an internet. Again, its specific implementation is not defined in the standard.

The last bit used is the *reliability bit* (*R bit*), which allows a user to request high reliability for the datagram.

The next two bits, 6 and 7, are not used at this time.

The *TOS field* is not used in some vendors' implementations of IP. Nonetheless, it will be used increasingly in the future as the internet capabilities are increased. For example, it is used in the Open Shortest Path First (OSPF) protocol discussed in Chap. 8. Consequently, a user should examine this field for future work and ascertain a vendor's use or intended support of this field.

The *total length* field specifies the total length of the IP datagram. It is measured in octets and includes the length of the header and the data. IP subtracts the header length field from the total length field to compute the size of the data field. The maximum possible length of a datagram is 65,535 octets (2^{16}). Gateways that service IP datagrams are required to accept any datagram that supports the maximum size of a PDU of the attached networks. Additionally, all gateways must accommodate datagrams of 576 octets in total length.

The IP protocol uses three fields in the header to control datagram fragmentation and reassembly. These fields are the *identifier*, *flags*, and *fragmentation offset*. The identifier field is used to identify uniquely all fragments from an original datagram. It is used with the source address at the receiving host to identify the fragment. The flags field contains bits to determine if the datagram may be fragmented. If it can be fragmented, one of the bits can be set to determine if this fragment is the last fragment of the datagram. The fragmentation offset field contains a value which specifies the relative position of the fragment to the original datagram. The value is initialized as 0 and is subsequently set to the proper number if/when the gateway fragments the data. The value is measured in units of eight octets. We will devote a special section later in this chapter to fragmentation/reassembly and the use of these three fields.

The *time-to-live* (*TTL*) parameter is used to measure the time a datagram has been in the internet. It is quite similar to CLNP's life-time field. Each gateway in the internet is required to check this field and discard it if the TTL value equals 0. A gateway is also required to decrement this field in each datagram it processes. In actual implementations, the TTL field is a number of hops value. Therefore, when a datagram proceeds through a gateway (hop), the value in the field is

decremented by a value of one. Implementations of IP may use a time-counter in this field and decrement the value in one-second decrements.

The TTL field is used not only by the gateway to prevent endless loops, but it can also be used by the host to limit the lifetime that segments have in an internet. Be aware that if a host is acting as a gateway, it must treat the TTL field by the gateway rules. Check with the vendor to determine when a host throws away a datagram based on the TTL value. Ideally, the TTL value could be configured and its value assigned based on observing internet performance. Additionally, network management information protocols such as those residing in SNMP might wish to set the TTL value for diagnostic purposes. Finally, if your vendor uses a fixed value that cannot be reconfigured, make certain it is fixed initially to allow for your internet's growth.

The *protocol* field is used to identify the next-level protocol above the IP that is to receive the datagram at the final host destination. It is quite similar to the type field found in the Ethernet frame. The Internet standards groups have established a numbering system to identify the most widely used upper layer protocols. Table 5.2 lists and describes these protocols.

The *header checksum* is used to detect a distortion that may have occurred in the header. Checks are not performed on the user data stream. Some critics of IP have stated that the provision for error detection in the user data would allow the gateway at least to notify the sending host that problems have occurred. (This service is indeed provided by a companion standard to IP, called the ICMP, which is discussed in Chap. 6.) Whatever one's view on the issue, the current approach keeps the checksum algorithm in IP quite simple. It does not have to operate on many octets, but it does require that a higher-level protocol at the receiving host perform some type of error check on the user data if it cares about its integrity.

IP carries two addresses in the datagram. These are labeled *source* and *destination addresses* and remain the same value throughout the life of the datagram. These fields contain the internet addresses examined in Chap. 3.

The *options* field is used to identify several additional services. As we shall see later, it is similar to the option part field of CLNP. The options field is not used in every datagram. The majority of implementations use this field for network management and diagnostics.

Figure 5.4 illustrates the format of the option field. Table 5.3 contains the values that are currently defined in the standard.

The options field length is variable because some options are of variable length. Each option contains three fields. The first field is coded

TABLE 5.2 Internet Protocol Numbers

Decimal	Key word	Protocol
0		Reserved
1	ICMP	Internet Control Message Protocol
2	IGMP	Internet Group Management Protocol
3	GGP	Gateway-to-Gateway Protocol
4		Unassigned
5	ST	Stream
6	TCP	Transmission Control Protocol
7	UCL	UCL
8	EGP	Exterior Gateway Protocol
9	IGP	Interior Gateway Protocol
10	BBN-MON	BBN-RCC Monitoring
11	NVP-II	Network Voice Protocol
12	PUP	PUP
13	ARGUS	ARGUS
14	EMCON	EMCON
15	XNET	Cross Net Debugger
16	CHAOS	Chaos
17	UDP	User Datagram Protocol
18	MUX	Multiplexing
19	DCN-MEAS	DCN Measurement Subsystems
20	HMP	Host Monitoring Protocol
21	PRM	Packet Radio Monitoring
22	XNS-IDP	XEROX NS IDP
23	TRUNK-1	Trunk-1
24	TRUNK-2	Trunk-2
25	LEAF-1	Leaf-1
26	LEAF-2	Leaf-2
27	RDP	Reliable Data Protocol
28	IRTP	Internet Reliable TP
29	ISO-TP4	ISO Transport Class 4
30	NETBLT	Bulk Data Transfer
31	MFE-NSP	MFE Network Services
32	MERIT-INP	MERIT Internodal Protocol
33	SEP	Sequential Exchange
34-60		Unassigned
61		Any host internal protocol
62	CFTP	CFTP
63		Any local network
64	SAT-EXPAK	SATNET and Backroom EXPAK
65	MIT-SUBN	MIT Subnet Support
66	RVD	MIT Remote Virtual Disk
67	IPPC	Internet Plur. Packet Core
68		Any distributed file system
69	SAT-MON	SATNET Monitoring
70		Unassigned

TABLE 5.2 Internet Protocol Numbers (Continued)

Decimal	Key word	Protocol
71	IPCV	Packet Core Utility
72-75		Unassigned
76	BRSAT-MON	Backroom SATNET Monitoring
77		Unassigned
78	WB-MON	Wideband Monitoring
79	WB-EXPAK	Wideband EXPAK
80-254		Unassigned
255		Reserved

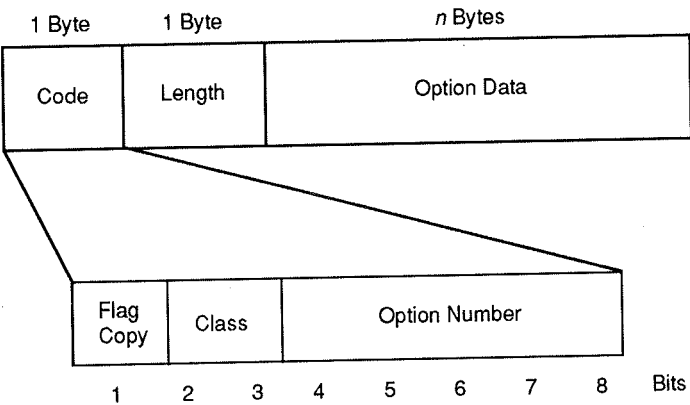


Figure 5.4 The IP Option Field.

TABLE 5.3 Option Codes

Class	Number	Length	Description
0	0	0	End of option list
0	1	0	No operation
0	2	11	Security
0	3	var	Loose source routing
0	7	var	Record route
0	8	4	Stream ID (obsolete)
0	9	var	Strict source routing
2	4	var	Internet timestamp

as a single octet containing the option code. The option code also contains three fields. Their functions are as follows:

<i>Flag copy (1 bit)</i>	0 = Copy option into only the first fragment of a fragmented datagram. 1 = Copy option into all fragments of a fragmented datagram.
<i>Class (2 bits)</i>	Identifies the option class (Table 5.3).
<i>Option Number</i>	Identifies the option number (Table 5.3).

The option class can be set to the following values:

- 0: A user datagram or a network control datagram
- 1: Reserved
- 2: Diagnostics purposes (debugging and measuring)
- 3: Reserved

The next octet contains the length of the option. The third field contains the data values for the option. The option field is described in more detail in the next section of this chapter.

The *padding* field may be used to make certain that the datagram header aligns on an exact 32-bit boundary.

Finally, the *data* field contains the user data. IP stipulates that the combination of the data field and the header cannot exceed 65,535 octets.

Major IP Services

This section provides an overview of the major services of IP. Be aware that vendors have different products for IP and some of them may not support all the features described in this section.

Internet header check routine

When a gateway receives the datagram, it checks the header to determine the type of traffic it is processing. If the traffic is an internet datagram, it passes the datagram to the internet header check routine. This module performs a number of editing and validity tests on the IP datagram header. The following checks are performed on the header:

- Valid IP header length
- Proper IP version number

Valid IP message length

Valid IP header checksum

Nonzero time to live field validity

If checks are performed and not passed, the datagram is discarded. If the checks are performed and passed, the internet destination address is examined to determine (a) if the datagram is addressed to this gateway, or (b) if the datagram is destined for another gateway. If it is not destined for this gateway, the datagram is passed to the IP forwarding routine for further routing.

IP source routing

IP can use a mechanism called *source routing* as part of its routing algorithm. Source routing allows an upper layer protocol (ULP) to determine how the IP gateways route the datagrams. The ULP has the option of passing a list of internet addresses to the IP module. The list contains the intermediate IP nodes that are to be transited during the routing of the datagrams to the final destination. The last address on the list is the final destination of an intermediate node.

When IP receives a datagram, it uses the addresses in the source routing field to determine the next intermediate hop. As illustrated in Fig. 5.5, IP uses a pointer field to learn about the next IP address. If a check of the pointer and length fields indicates that the list has been completed, the destination IP address field is used for routing. If the list is not exhausted, the IP module uses the IP address indicated by the pointer.

The IP module then replaces the value in the source routing list with its own address. Of course, it must then increment the pointer by one address (4 bytes) for the next hop to retrieve the next IP address in the route. With this approach, the datagram follows the source route dictated by the ULP and also records the route along the way.

Figure 5.5 provides an example of the use of route recording. In the first step, IP uses the pointer to locate the next address in the route data field. In this example, it locates address 128.2.3.4 and makes a routing decision based on this address. In the second step, it places its own address in the route data field in the same location of the current destination address. In the third step, it increments the pointer value to enable the next IP module to determine the next (or final) hop in the route.

Routing operations. The IP gateway makes routing decisions based on the routing list. If the destination host resides in another network, the IP gateway must decide how to route to the other network. Indeed, if

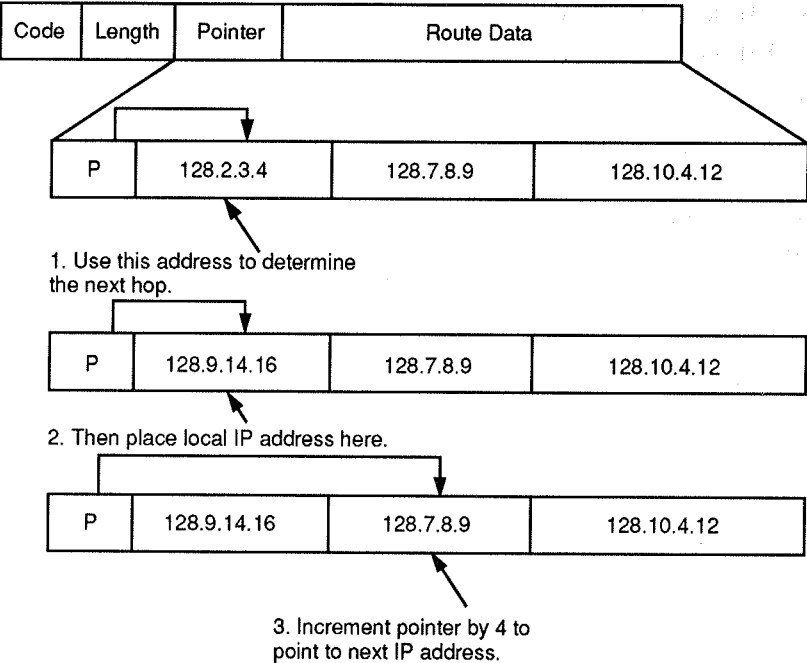


Figure 5.5 Source Routing.

multiple hops are involved in the communications process, then each gateway must be traversed and the gateway must make decisions about the routing.

Each gateway maintains a routing table that contains the next gateway on the way to the final destination network. In effect, the table contains an entry for each reachable network. These tables could be static or dynamic, although dynamic tables are more common. The IP module makes a routing decision on all datagrams it receives.

The routing table contains an IP address for each reachable network and the address of a neighbor gateway (that is, a gateway directly attached to this network). The neighbor gateway is the shortest route to the destination network. Otherwise, the IP gateway logic establishes that the gateway is directly connected to this network.

The IP routing is based on a concept called the *distance metric*. This value is usually nothing more than the fewest number of hops between the gateway and the final destination. The gateway consults its routing table and attempts to match the destination network address contained in the IP header with a network entry contained in the routing table. If no match is found, it discards the datagram and builds an ICMP message to send back to the IP source. This message would contain a "destination unreachable" code. If a match is found in the routing table, the gateway then uses it to determine the outgoing port.

Some implementations of TCP/IP have allowed a host to perform source route forwarding and to act as an intermediate hop through the full route. If the reader uses a system in which a host performs source routing, the host should adhere to all the rules of a conventional gateway in managing source routed datagrams. Again, this point is made to emphasize that the task of the host in any type of gateway function should be commensurate with the gateway, and the software should not be scaled down for purposes of efficiency. For example, the TTL field must be decremented by the host; and the host should be able to generate the ICMP destination unreachable messages in case the source route fails or in case fragmentation cannot be performed. It must also be able to perform the timestamp option (discussed shortly) in accordance with the proper rules of an IP gateway.

Loose and strict routing. IP provides two options in routing the datagram to the final destination. The first, called *loose source routing*, gives the IP modules the option of using intermediate hops to reach the addresses obtained in the source list as long as the datagram traverses the nodes listed. Conversely, *strict source routing* requires that the datagram travel only through the networks whose addresses are indicated in the source list. If the strict source route cannot be followed, the originating host IP is notified with an error message. Both loose and strict routing require that the route recording feature be implemented.

Route recording option

The route recording option operates in the same manner as source routing, with the recording feature which was just discussed. This means that any IP module that receives a datagram must add its address to a route recording list. The idea is illustrated in Fig. 5.6, except that this option is not using addresses in the recording field to determine the next hops. For the route recording operation to occur, the receiving IP module uses the pointer and length fields to determine if any room is available to record the route. If the route recording list is full, the IP module simply forwards the datagram without inserting its address. If it is not full, the pointer is used to locate the first empty full octet slot, the address is inserted, and the IP module then increments the pointer to the next IP slot.

The timestamp option

Another very useful option in IP is the provision for timestamping the datagram as it traverses each IP module through the internet. This idea allows a network manager not only to determine the route of the

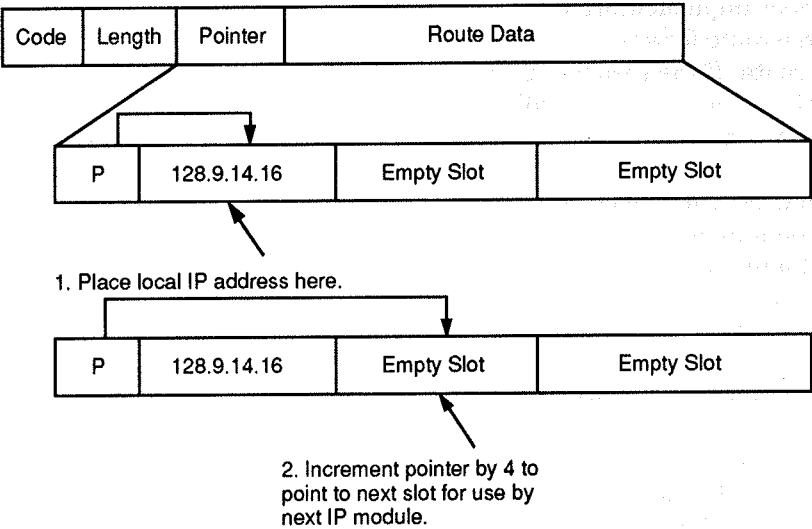


Figure 5.6 Route Recording.

datagram through the internet but also the time at which each IP module processed the datagram. This can be quite useful in determining the efficiency of gateways and routing algorithms.

The format for the options fields for timestamp operations is shown in Fig. 5.7. As with previous options, the length and pointer fields are used to identify the proper slot in which to place an IP address *and* the timestamp related to this address. Therefore, the pointer increments itself across an IP address and the timestamp for the address. The *oflw* field is only used when an IP module cannot register a timestamp due to (for example) lack of resources or an option field that is too small. This value is incremented by each module that encounters this problem.

A 4-bit flags field is used to provide guidance to each IP module about the timestamp operations. The values currently described are as follows:

- 0: Timestamps are to be recorded only and are stored in consecutive 32-bit words.

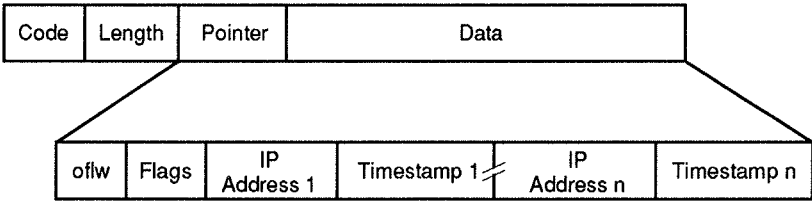


Figure 5.7 The Timestamp Option.

1: Each timestamp is to be preceded by the IP address of the relevant module.

3: IP addresses are already specified by the originator, and the gateway is tasked with recording the timestamp in its relevant IP address area.

The time used with the timestamp is based on milliseconds (ms) using universal time (the old Greenwich mean time). Obviously, the use of the universal time does not guarantee completely accurate timestamps between machines, because machines' clocks may vary slightly. Nonetheless, in most networks the universal time in milliseconds provides a reasonable degree of accuracy.

Fragmentation and reassembly

An IP datagram may traverse a number of different networks that use different PDU sizes, and all networks have a maximum PDU size, called the *maximum transmission unit (MTU)*. Therefore, IP contains procedures for dividing (fragmenting) a large datagram into smaller datagrams. It also allows the ULP to stipulate that fragmentation may or may not occur. Of course, it must also use a reassembly mechanism at the final destination which places the fragments back into the order originally transmitted.

When an IP gateway module receives a datagram which is too big to be transmitted by the transit subnetwork, it uses its fragmentation operations. It divides the datagram into two or more pieces (with alignment on 8-octet boundaries). Each of the fragmented pieces has a header attached containing identification, addressing, and, as another option, all options pertaining to the original datagram. The fragmented packets also have information attached to them defining the position of the fragment within the original datagram, as well as an indication if this fragment is the last fragment.

Referring to Fig. 5.3, the flags (the 3 bits) are used as follows:

Bit 0 = reserved

Bit 1: 0 = fragmentation and 1 = do not fragment

Bit 2 (M bit): 0 = last fragment and 1 = more fragments

Interestingly, IP handles each fragment operation independently. That is, the fragments may traverse different gateways to the intended destination, and they may be subject to further fragmentation if they pass through networks that use smaller data units. The next gateway uses the offset value in the incoming fragment to determine the offset values of fragmented datagrams. If further fragmentation is done at another gateway, the fragment offset value is set to the loca-

tion that this fragment fits relative to the original datagram and not the preceding fragmented packet.

Figure 5.8 shows an example of multiple fragmentation operations across two gateways. Be aware that this example shows the *OL* (*overall length*) field. This field is not carried in the datagram but is used here to assist the reader in this analysis.

Subnet 128.3 uses a 1500-octet PDU size. It passes this data unit to gateway A. Gateway A decides to route the PDU to subnet 21.4, which supports a 512-octet PDU size. The gateway fragments the 1500 data

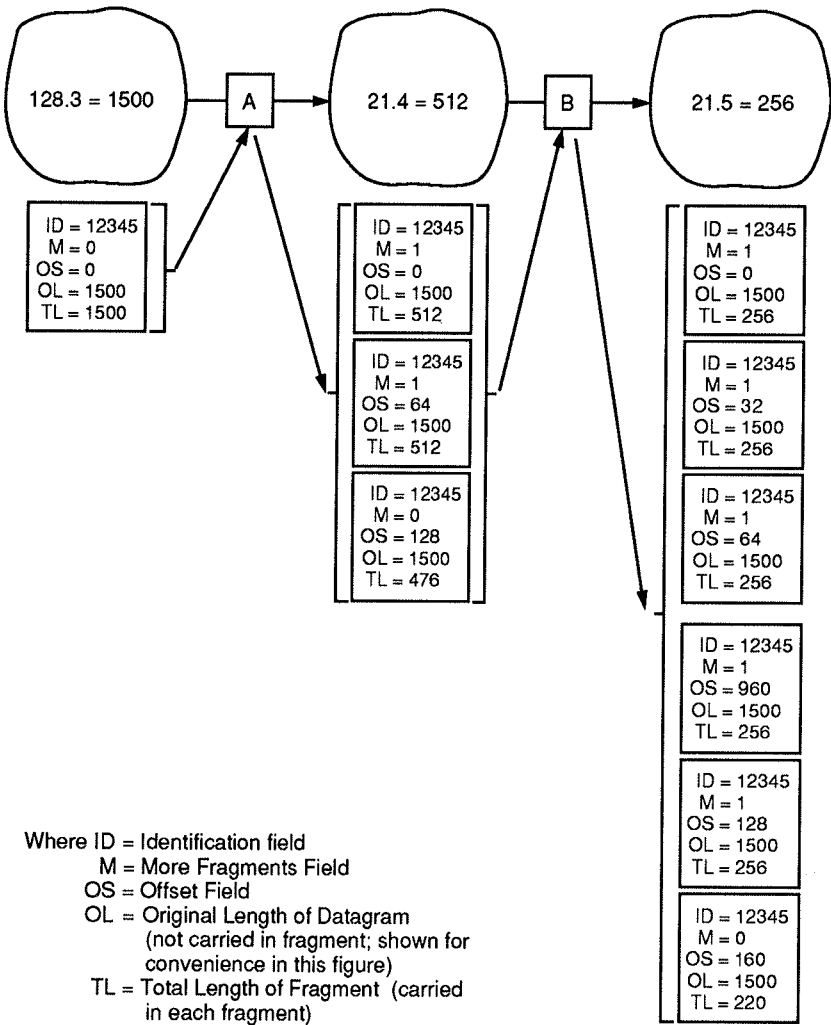


Figure 5.8 Fragmentation Operations at the Gateways.

unit into three smaller data units of 512, 512, and 476 octets. Thus, $1500 = 512 + 512 + 476$. The last segment containing 476 octets is filled (padded) with zeros to equal a total of a multiple of 8. Therefore, this data field is 480 ($476 + 4$, which is an even multiple of 8).

Gateway A passes the data to subnetwork 21.4, which delivers it to gateway B. This gateway decides that datagram fragments are to be delivered to subnetwork 21.5. Since the gateway knows that this network uses a PDU size of 256 octets, it performs further fragmentation. It divides the 512 octet fragments into yet smaller data units and, using the offset values in the three incoming fragments, adjusts accordingly the offset values in the outgoing data units. Notice that the offset values are reset at gateway B, and their values are derived from the offset values contained in the preceding fragments.

As illustrated in Fig. 5.9, reassembly occurs at the receiving host. The IP module sets up buffer space when the first fragment is re-

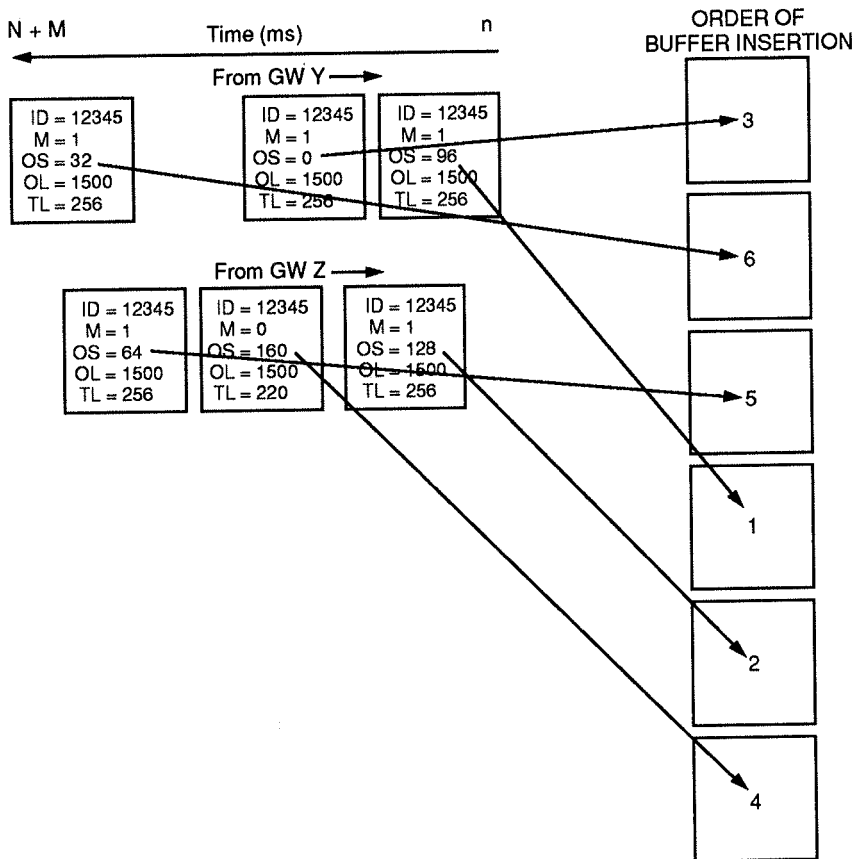


Figure 5.9 Reassembly of the Fragments.

ceived. A buffer is reserved for each fragment, and the fragment is placed in an area within the buffer relative to its position in the original datagram. As the fragments arrive, they are placed into the proper location in the buffer (pigeon holing). When all the fragments have been received, the IP module passes the data to the ULP in the same order that it was originally sent from the sending ULP.

For purposes of discussion, we assume that the datagram fragments depicted in Fig. 5.8 are routed to other gateways, say gateways Y and Z. To continue the analysis, Fig. 5.9 shows that the fragmented datagrams arrive from gateways Y and Z in the order depicted by the "time arrow," with earliest arrival at time n and the latest time of arrival at time $n + m$. Therefore, the fragments arrive in the following order (we are using the offset values in the figure to identify the fragment):

- First:* Fragment with offset value of 96
- Second:* Fragment with offset value of 128
- Third:* Fragment with offset value of 0
- Fourth:* Fragment with offset value of 160
- Fifth:* Fragment with offset value of 64
- Sixth:* Fragment with offset value of 32

The receiving machine has a rather easy job of figuring out where the fragments are to be placed. The IP module simply multiplies the offset value by 8 to determine which slot in the buffer is to receive the fragment. For example, the first arriving fragment's relative position in the buffer is computed as $96 * 8 = 768$, or memory address 768. (If the reader wishes to test the calculations, we are using position 0, not 1, as the first position.)

In Fig. 5.9, the reassembling host initially does not know the length of the complete IP datagram until it receives the fourth fragment, which contains the $M = 0$ (no more fragments), the offset value, and the fragment length. Since the offset is 160 and the length is 220 octets, it now knows that the total datagram is (160 offset values * 8 octets per value + 220 octets in final fragment = 1500 octets).

We now see why the M bit is so important. Since the length field in the fragment does not refer to the size of the original datagram but to the size of the fragment, the only method to determine the original length (and the final fragment) is the $M = 0$ indicator.

In the event that some fragments do not arrive or have been discarded because they exceeded the TTL parameter, IP will discard the fragments of the partially reassembled datagram. In addition, the receiving computer turns on a reassembly timer upon detection of the

arrival of the first fragment. This timer, which is set by the network manager, can be used to ensure that all fragments arrive in a timely manner. If the timer times out before all fragments have arrived, the received fragments are discarded. If a user does not want fragmentation to occur, the fragment flag can be turned to 1, which indicates that fragmentation must not occur. This may be desirable if fragmentation creates excessive overhead because the reassembly timer continues to discard fragments which require retransmission from higher-layer protocols. However, this situation must be weighed against the fact that turning on the "don't fragment" flag means that datagrams will be discarded by gateways if the MTU exceeds the size of the subnetwork capability.

Reassembly at the gateways? Connectionless gateways using dynamic routing (such as IP) do not reassemble datagrams. The task is impossible because all fragments belonging to the original datagram may not be processed by the same gateways. As a consequence, the gateway does not know how to compute the offset values for fragments it does not receive. In contrast, connection-oriented gateways can perform intermediate reassembly because, by the very nature of the system, all PDUs pass through the same gateway(s). For example, imagine in Fig. 5.8 that gateway A is a connection-oriented gateway. It could reassemble the fragments for transmission over a higher-capacity subnetwork by detecting the $M = 0$ in the fragment with an offset value of 128 and a length of 476 ($128 \text{ offset values} * 8 \text{ octets per offset} + 476 \text{ octets in final fragment} = 1500 \text{ octets}$).

Retain fragments to await retransmission of other fragments? Several of this writer's clients have expressed concern about IP discarding fragments based on the expiration of the fragmentation timer. In some instances, this can create a significant amount of replicated traffic on a network. This brief discussion should shed some light on the problem. First, we assume that discarded fragments are retransmitted by a higher-layer protocol (e.g., TCP). The traffic is fragmented (once again) by the originator and sent to the receiver.

The retransmitted fragments would contain the same values in the IP identification field as the original submission. Assuming that the receiver's software does not discard the fragments that were acceptable in the original transmission, one could argue that the identification field could be used to reconstruct a complete data unit by using the fragments of the retransmissions. Of course, this also means that if a pigeon hole is full, the received duplicate fragment would be discarded.

A brief analysis shows that it is not worth the effort. Filling vacant

pigeon holes with retransmitted fragments is almost pure chance because some of the retransmitted fragments might undergo different fragmentation operations since the fragments were routed differently the second time. It is especially onerous if the transmitting end must determine which fragments should be transmitted and which should not. Indeed, making IP this smart moves it toward connection-oriented features—which should be avoided.

Also, be aware that the reassembly time-out is based on the remaining value of the TTL field in the IP header. This value does not work very well because the vast majority of TTL values are implemented with a hop-count metric and not a time metric. Therefore, an implementation of the reassembly time-out should be based on an actual clock.

Before leaving the subject of the options in IP, it should prove useful to reexamine the flag copy feature (see Fig. 5.4). A network administrator should determine if the flag copy should be used for certain options. For example, is it useful for all fragments to have the route recording option? After all, if these fragments move through different routes in an internet, the receiver might have difficulty in determining a single list of routes. On the other hand, such a feature still does not preclude examining the routes to determine the efficiency of the internet gateways and IP modules. However, for purposes of simplicity it may make sense (and the IP standard so requires) to stipulate that a route recording option is only used in one of the fragments of a fragmented datagram.

Next, consider when the option should be copied into all fragments. One that comes to mind is source routing. If the user wants source routing to be applied to a datagram, it is logical to assume the user wants it to be applied to all fragments of a fragmented datagram.

IP Address and Routing Tables

Address table

The Internet Activities Board (IAB) has published in RFC 1213 the definition for the IP address table. This table is depicted in Fig. 5.10. It consists of five columns and 1 to n rows. Each row pertains to an IP address at an entity (a host, gateway, etc.). There may be more than one row entry if the machine has more than one IP address. The contents of this table are as follows:

The *Entry Address* contains the IP address for this entry's interface.

The *Entry IfIndex* contains the interface number (port number) pertaining to this entity's connection to a subnetwork.

	Entry Address	Entry If Index	Entry Net Mask	Broadcast Address	Maximum Size of Datagram
Address Entry 1					
Address Entry 2					
// Address Entry <i>n</i>					

Figure 5.10 The IP Address Table.

The *Entry Net Mask* column contains the subnet mask associated with the IP address in the entry address column.

The *Broadcast Address* contains a value for the least significant bit in the IP broadcast address. It is used for sending datagrams on the local interface and is associated with the IP address of this row.

The *Maximum Size of Datagram* represents the maximum size of a datagram that can be processed by this IP module.

Routing table

Until 1990, an IP routing table was designed based on an individual vendor's perception of the need for the entries in the table. With the publication of the Internet Management Information Base (MIB), a more formal definition of the IP routing table is now available. Figure 5.11 shows the IP routing table as defined in the MIB standard published as RFC 1213.

Each row of the IP routing table contains an entry for each route that is known to the IP module storing this table. The columns represent the information available on each route. A brief description of each column follows.

The *Destination* entry contains the IP address of the destination for this route. If this column is coded as 0.0.0.0, the route is considered a default route.

The *IfIndex* entry stands for the interface index. It identifies the local interface (more commonly known as a physical port) through which the next hop in the route can be reached.

	Destination	If Index	Metric 1 // Metric 5	Next Hop	Route Type	Routing Protocol	Route Age	Routing Mask	Route Information
Route 1									
Route 2									
Route 3									
// Route n									

Figure 5.11 The IP Routing Table.

The next five columns are labeled *Metric*. The metric entries contain information about the cost metric used for determining the route. With most systems, the cost metric is the number of hops to reach the destination. However, with the evolution of more sophisticated route discovery protocols (such as OSPF), it is possible that more than one cost metric may be used in calculating the route. As the table indicates, up to five cost metrics can be stored.

The *Next Hop* entry contains the IP address identifying the next hop for this route.

The next column in the table is the *Route Type*. It is set to one of four values to provide the following information: 1 = none of the following, 2 = an invalid route, 3 = a directly connected route (a directly connected subnet), and 4 = indirect (an indirect connection to reach this destination).

The next column is labeled *Routing Protocol*. This entry identifies the route discovery protocol by which this route was learned. If the route is discovered with RIP, the value in this column must be 8; if it is discovered with the Cisco internal group routing protocol, it must be set to 11; and so on.

The *Route Age* column is a value, in seconds, since the route was updated or verified.

The next field in the table contains the *Route Mask* for this route. Remember that this mask is logical ANDed with the destination address in the IP datagram before being compared to column 1 in this table (labeled *Destination*).

The last column, labeled *Route Information*, allows a reference to an MIB definition pertaining to a particular routing protocol. Its value depends on the type of routing protocol used in this row entry.

As an example of a row entry into the IP routing table, please refer back to Chap. 3, Fig. 3.10. This figure shows one of the destination networks as 14.0.0.0. Consequently, from the perspective of the routing table at gateway 9, the column entries for this network are as follows:

Destination = 14.0.0.0

Interface (port) Index = 3

Metric 1 = number of hops (the remaining metric columns are empty)

Next Hop = 13.0.0.2

Route Type = indirect (4)

Routing Protocol = egp (5)

Route Age = 5 seconds (for example)

Route Mask = several options

Route Information = value that depends on routing protocol entry

IP Service Definitions and Primitives

IP/ULP primitives

IP uses two primitives to define the services it provides to the adjacent ULP. The transmitting ULP utilizes the SEND primitive to request the services of IP. In turn, IP uses the DELIVER primitive to notify the destination ULP of the arrival of data. The interface of IP with the upper layer is quite simple, because IP is designed to operate with diverse ULPs.

These service definitions (implemented with primitives) are somewhat abstract to allow them to be tailored to the specific host operating system. For example, the SEND and RECV services on a UNIX-based system can be implemented with UNIX system library calls.

The relationship of IP and the ULP is shown in Fig. 5.12, which depicts host A sending data to host B. The arrows are reversed if host B sends data to host A.

The parameters associated with the primitives inform the ULP and IP about the operation requested from the ULP (SEND) and performed by IP (RECV).

The SEND parameters are also used to create the IP header. Con-

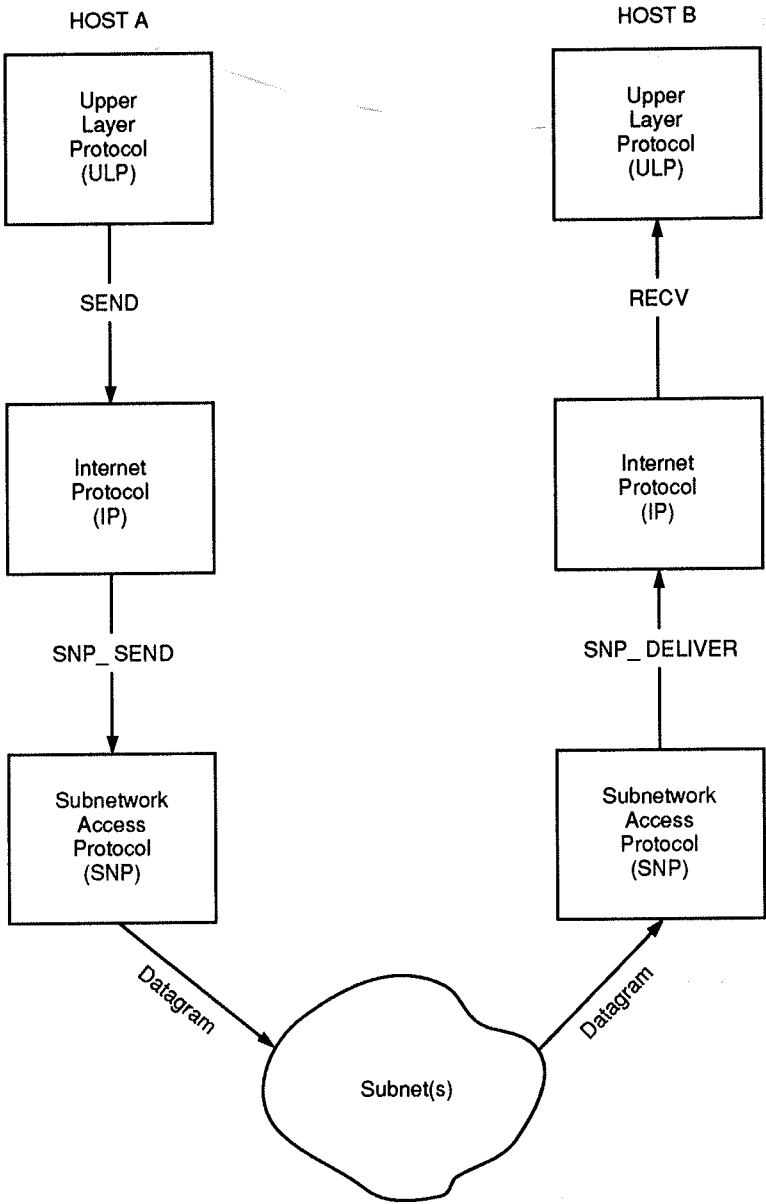


Figure 5.12 The IP Primitives.

sequently, they correlate closely to the IP header fields of a transmitted datagram, shown in Fig. 5.2. Likewise, the parameters in the RECV primitive correlate closely to the parameters of the received datagram. The SEND primitive contains the following information:

Source address: IP address of the host sending the data
Destination address: IP address of host to receive the data
Protocol: Name of the recipient ULP (e.g., TCP, UDP, etc.)
TOS indicators: precedence, reliability, delay, and throughput
Identifier: Optional, for fragmentation control
Don't fragment indicator: Yes or no
Time to live: In number of hops
Data length: Length of data being transmitted (0, if no data)
Buffer pointer: Pointer to the datagram
Option data: Options requested by a ULP (security, loose or strict source routing, record routing, stream identification, or timestamp)
Data: Present when data length is greater than 0
Result: Result of this SEND request:
 Datagram sent
 Error in arguments or network error

The RECV primitive contains the following information:

Source address: IP address of sending host
Destination address: IP address of recipient host
Protocol: Name of the recipient ULP as supplied by the sending ULP (e.g., TCP, UDP, etc.)
TOS indicators: Relative transmission quality associated with unit of data: precedence, reliability, delay, and throughput
Data length: Length of received data
Buffer pointer: Pointer to the datagram
Option data: Options requested by source ULP (security, loose or strict source routing, record routing, stream identification, or timestamp)
Data: Present when data length is greater than 0
Result: Result of this RECV request

IP/SNP primitives

The primitives between IP and the subnetwork access protocol (SNAP) are similar to the IP/ULP primitives. As shown in Fig. 5.12, two primitives are invoked for the IP/SNP operations. The `SNP_SEND` primitive is used by IP to invoke a service of the SNP; the `SNP_DELIVER` is used by the SNP to deliver a datagram to the IP module. The `SNP_SEND` primitive contains the following information:

Local destination address: Subnetwork address of the destination

TOS indicators: precedence, reliability, delay, and throughput

Length of datagram

The datagram

The `SNP_DELIVER` primitive contains the datagram and error indicators (the latter is optional and not defined in the standard).

Other IP/SNP service definitions. Many IP modules rest on top of IEEE LANs and interface with IEEE 802.2 (logical link control, LLC). Chapter 3 (Fig. 3.5) explains the method by which LLC and IP use the LSAP header extension to coordinate their operations. However, as of this writing, no RFC exists to explain the relationship of the IEEE 802.2 LLC primitives and IP. The next section provides this explanation.

Network layer/LLC primitives

The IP/SNP primitives explained in the previous section are not designed for the LLC interface. However, if an implementation only makes use of LLC connectionless data transfer service, the interface is simple. A brief explanation of this service follows (see Fig. 5.13).

Two primitives are used for connectionless data transfer. They are passed between LLC and its upper layer protocol (which could be IP or some other module).

DL-UNITDATA.request (source-address, destination-address, data, priority)

DL-UNITDATA.indication (source-address, destination-address, data, priority)

The request primitive is passed from the network layer to LLC to request that a link service data unit (LSDU) be sent to a remote link service access point. The address parameters are equivalent to a combination of the LLC SAP and MAC addresses. The priority field is passed to MAC and implemented (except for 802.3, which has no priority mecha-

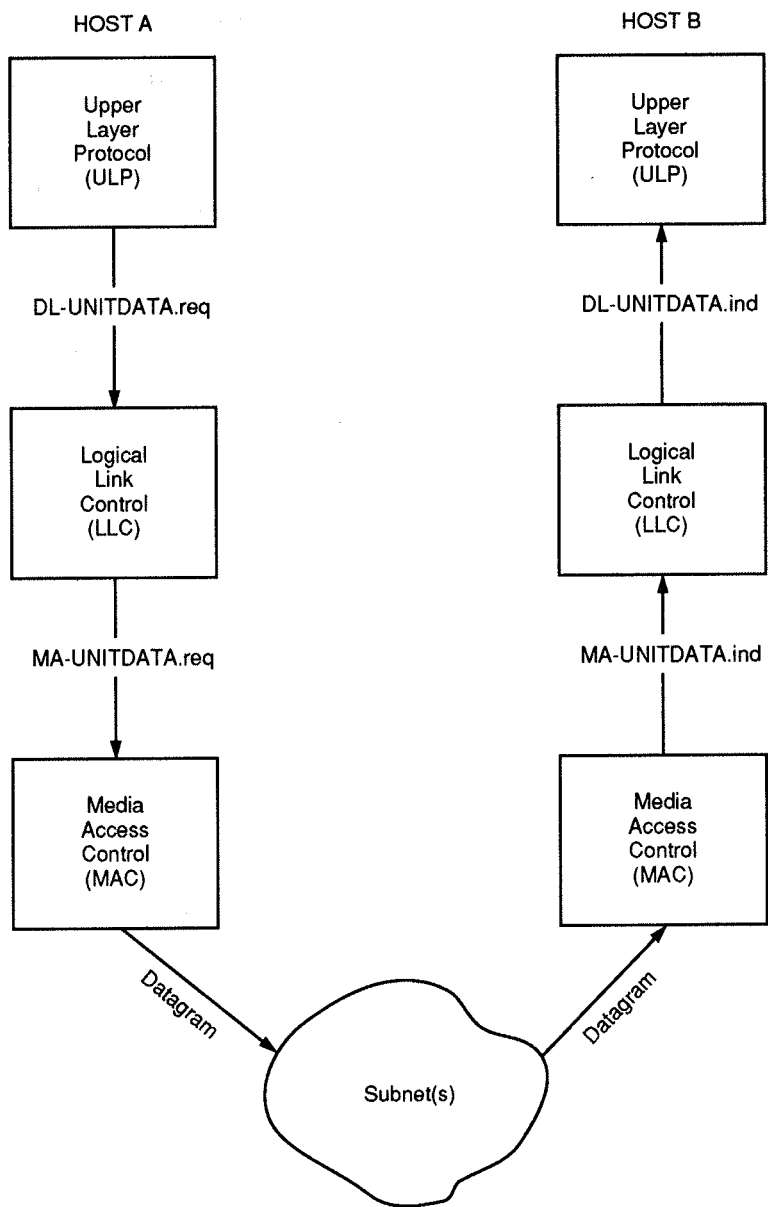


Figure 5.13 IP Over LLC in an IEEE LAN.

nism). The indication primitive is passed from LLC to the network layer to indicate the arrival of an LSDU from a remote entity.

A comparison of Figs. 5.12 and 5.13 reveals close similarities between IP/SNP and ULP/LLC operations, but differences exist in the parameters associated with the IP/SNP and ULP/LLC primitives.

If IP is placed on top of LLC, a convergence protocol must be implemented to map the SNP_SEND primitive and its parameters to the DL-UNITDATA.request and its parameters. The protocol must also map the SNP_DELIVER primitive and its parameters to the LLC DL-UNITDATA.indication and its parameters. The address fields present no major problem if RFC 1042 is followed (see Chap. 2, "Extension of LSAP Header"). A potential problem is deciding how to handle the TOS parameters. Perhaps the best approach is to ignore them at the LLC and MAC levels, since they are mostly irrelevant for use in a high-capacity LAN.

Multicasting

Like the physical layer, the internet network layer allows the use of multicasting PDUs. The concept is quite similar to LAN multicasting, in which a single PDU (in this case an IP datagram) is sent to more than one host. The set of hosts forms a *multicast group*.

IP multicasting allows hosts to operate on one or multiple physical networks. A host may belong to more than one multicast group, it may belong to a permanent multicast group (a *well-known* group), or it may dynamically enter and leave groups as the need dictates. Also, a host does not have to be a member of the multicast group to send data to the group.

Multicasting is accomplished through the use of the IP class D address discussed in Chap. 3. The first 4 bits of the 32-bit address field are set to 1110; the next 28 bits identify the specific multicast group. Due to the structure of the address space the classes of A, B, and C addresses are meaningless when using multicasting.

The permissible ranges for multicast addresses are from 224.0.0.0 through 239.255.255.255. The Internet does not allow the address space of 224.0.0.0 to be used, and the address space of 224.0.0.1 is reserved for an *all hosts* group. An all hosts group identifies all hosts and gateways that are participating in an internet IP multicast operation.

Figure 5.14 shows an example of how IP multicasting operates. Gateway 1 (G1) receives an IP datagram from another part of an internet. The destination IP address is a multicast address (a source address is not allowed to contain multicast values, for obvious reasons). The gateway is responsible for interpreting the multicast address and forwarding it to the proper hosts (for which it has authority)

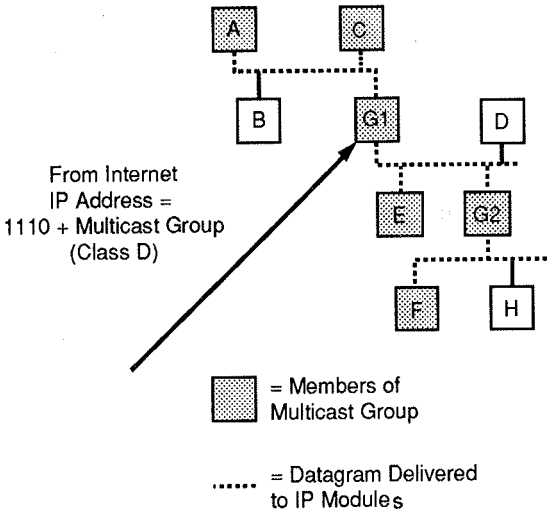


Figure 5.14 Multicasting Operations.

and to other participating multicast gateways (G2). In this example, hosts A, C, and E are members of the same multicast group. G2 and host F are also members.

One approach to sending this traffic to the proper host on a LAN is to map the IP multicast address to an Ethernet multicast address. The mapping is quite simple and proceeds as follows: The low-order 23 bits of the IP address are mapped into the low-order 23 bits of the Ethernet multicast address. This approach is not perfect because a one-to-one relationship does not exist between the 28 bits of the IP address and the 23 bits of the Ethernet address. However, it is a reasonable solution because the chances are quite small that any two groups will choose the same 23 bits for the multicast address value. In any event, it is prudent to make certain that the receiving software has the ability to discard datagrams that are not appropriate.

Hosts can be configured to either send or receive multicast datagrams. Alternately, hosts can be configured to send but not receive multicast datagrams or not to participate in any multicast operations.

The implementation of multicasting must ensure that the software is capable of allowing a host to join or leave a multicast group. A means must be devised to allow hosts to inform other hosts about the relationships and status of memberships. The IP software must keep bookkeeping operations about which host belongs to separate multicast groups and delete and add them accordingly. These administrative tasks are accomplished through the Internet Group Management Protocol (IGMP), which is discussed in the next section.

Internet Group Management Protocol (IGMP)

The IGMP allows gateways and hosts to inform each other about multicasting operations, such as joining and leaving multicast groups. Additionally, it allows the machines to communicate status messages about multicasting information. IGMP works in a manner quite similar to the Internet Control Message Protocol (ICMP). An IP datagram is used to transport the IGMP message between the machines. Be aware that IGMP is required for all machines that are going to implement IP multicasting.

IGMP also has some similarities to a LAN token bus protocol in how a machine is added as a member to a group. First, when a host wishes to join a multicast group, it must send out a message (just as, in a token bus, a LAN station sends out a set solicitor message). This message is an all-host multicast address in which the host establishes that it is a member of a multicast group. The IGMP message is received by the local multicast gateway that performs the necessary operations to route the traffic to other multicast gateways throughout an internet.

Once these operations have been performed, the multicast gateway must periodically query the host to determine if a host is still a member of a group. If the query does not get a response from the host, then the multicast gateway is required to stop advertising membership for that host to other multicast gateways.

The format for the IGMP message is quite simple and is illustrated in Fig. 5.15. The message consists of five fields.

The *version* field gives the current protocol version, which at this time is 1.

The *type* field identifies the field as a query with a value of 1 or a response with a value of 2.

Logically enough, the *unused* field is not used.

Version (4)
Type (4)
Unused (8)
Checksum (8)
Group Address (32)

(n) = number of bits in the field

Figure 5.15 The IGMP Message Format.

The *checksum* field is used to check for errors. The calculation uses the same algorithm as the IP and TCP protocols.

The *group address* field contains the group address for hosts to report their membership in the multicast group. This field has no meaning for a query message.

Other Thoughts on IP

The reader may be surprised at the brevity of the discussion of IP. We are not shirking our responsibility in the analysis; rather, we are merely reflecting the fact that IP does not contain a great number of functions. Indeed, one of the attractive aspects of IP is its simplicity and efficiency. However, as the reader may surmise, the supporting protocols to IP, such as TCP, and the addressing resolution and naming service protocols, discussed in earlier chapters, provide for a complex and rich internetworking environment.

It has been stated in earlier chapters that IP is one of the most widely used internetworking protocols in the world. Today, hundreds of products are available in the marketplace that use the IP architecture. Nonetheless, the use of IP will diminish gradually as users and vendors move to the ISO 8473 internetworking protocol.

The end user will most likely never know that the change has taken place if the implementation is well conceived. The network designer will see a similarity between the two protocols, but they are different enough that software must be written almost from scratch.

The next section introduces the ISO 8473 connectionless network protocol (CLNP).

Connectionless-Mode Network Service (ISO 8473)

ISO 8473 is a specification that describes the architecture for connectionless-mode network service. It is also called the connectionless network protocol (CLNP). It is quite similar to IP in the functions it performs, but it is not compatible with IP.

ISO 8473 has been designed to operate with the hosts attached to public and private networks and the gateways that service the hosts and networks. As will be seen, the protocol does not involve itself with the particular characteristics of the underlying subnetworks. Like IP, the basic idea is to demand little in the way of services from the subnetworks except to transport the PDU.

The protocol communicates by exchanging internetwork PDUs (IPDUs) in a connectionless (datagram) fashion. Each IPDU is treated

independently and does not depend on the state of the network for any particular establishment connection time (since no connection exists in a connectionless network). Routing decisions are made independently by each forwarding internetworking node. It is also possible for the source user to determine the routing by placing the routing information in the IPDU source routing field.

The ISO 8473 connectionless-mode network service is provided by two primitives: N-UNITDATA.request and N-UNITDATA.indication. These two primitives are quite similar to the primitives used by the IP. The reader should note that the protocol uses no connection and clear primitives, which is a rather obvious indication of its connectionless attributes. The two primitives each contain four parameters: NS-Source-Address, NS-Destination-Address, NS-Quality-of-Service, and NS-Userdata. All parameters are explained shortly.

The ISO 8473 PDU

A brief description of the fields of the 8473 PDU is provided in Table 5.4. As the reader reviews this material, the similarity to the IP datagram will be evident. Figure 5.16 shows the format of the data unit. The next sections discuss how the fields are used in 8473 operations.

Quality of service (QOS) functions

An underlying subnetwork may provide several quality of service (QOS) functions. The services are negotiated when the primitives are exchanged between layers. Of course, the primitive parameters must be based on *a priori* knowledge of the availability of the services within the subnetwork. It does little good to ask for a QOS feature if it is known that the subnetwork does not provide it. Be aware that the values of QOS apply to both ends of the network connections (NC), even though the NC may span several subnetworks that offer different services.

Use of ISO 8348. For QOS choices, ISO 8473 uses the following QOS functions described in ISO 8348:

Transit delay: Establishment of the elapsed time between a data request and the corresponding data indication. This QOS feature applies only to successful PDU transfers. The delay is specified by a desired value up to the maximum acceptable value. All values assume a PDU of 128 octets. User-initiated flow control are not measured in these values.

Residual error rate (RER): The ratio of total incorrect, lost, or duplicated PDUs to total PDUs transferred:

TABLE 5.4 Functions of Fields in 8473 PDU

Protocol identifier	Identifies the protocol as ISO 8473.
Length indicator	Describes the length of the header.
Version/protocol	Identifies the version of ISO 8473.
Lifetime	Represents the lifetime of the PDU. It is coded in units of 500 ms.
Segmentation permitted	Indicates if segmentation is permitted. The originator of the PDU determines this value, and it cannot be changed by any other entity.
More segments	Indicates if more user data is forthcoming. It is used when segmentation takes place. When the bit equals 0, it indicates that the last octet of the data in this PDU is the last octet of the user data stream (the network service data unit, NSDU).
Error report	Indicates that an error report is to be generated back to the originator if a data PDU is discarded.
Type code	Describes the PDU as a data PDU or an error PDU.
PDU segment length	Specifies the entire length of the PDU (header and data). If no segmentation occurs, the value of this field is identical to the value of the total length field.
Checksum	Calculated on the entire PDU header. A value of 0 in this field indicates that the header is to be ignored. A PDU is discarded if the checksum fails.
Destination and source addresses and address lengths	Since the source and destination addresses are variable in length, the length fields are used to describe their length. The actual addresses are Network Service Access Points (NSAPs).
Data unit identifier	Identifies an initial PDU to reassemble a segmented data unit correctly.
Segmentation offset	If the original PDU is segmented, this field specifies the relative position of this segment in relation to the initial PDU.
Total PDU length	Contains the entire length of the original PDU, which includes both the header and data. It is not changed for the lifetime of the PDU.
Options	Optional parameters are placed in this part of the PDU such as route recording, quality of service parameters, priorities, buffer congestion indication, padding characters, and designation of security levels.
Data	Contains the user data.

Protocol Identifier
Length Indicator
Version/Protocol ID Extension
Lifetime
Segment/More/Error Report/Type Code
Segment Length
Checksum
Destination Address Length
Destination Address
Source Address Length
Source Address
Data Unit Identifier
Segmentation Offset
Total PDU Length
Options
Data

Figure 5.16 The ISO 8473 PDU.

$$\text{RER} = \frac{N(e) + N(1) - N(x)}{N}$$

where RER = residual error rate; N(e) = PDUs in error; N(1) = lost PDUs; N(x) = duplicate PDUs; N = number of PDUs.

Cost determinants: A parameter to define the maximum acceptable cost for a network service. It may be stated in relative or absolute terms. Final actions on this parameter are left to the specific network provider.

Priority: Used to determine preferential service in the subnetwork. Outgoing transmission queues and buffers are managed based on the priority values contained in the PDU header options field.

Protection against unauthorized access: Used to direct the subnetwork to prevent unauthorized access to user data.

Protocol functions

ISO 8473 includes several optional or required protocol functions. Each function provides a specialized service to the network user. In a sense, the ISO 8473 protocol functions are similar to the QOS features, except that they are performed as an integral part of the protocol.

Traffic management between subnetworks

This section provides an illustration of how a connectionless-mode internetwork protocol transfers data between subnetworks. While different vendors use various techniques for the provision of connectionless service, many use the concepts described herein.

When the protocol receives the NS-Source-Address and NS-Destination-Address parameters in the N-UNITDATA.request primitive from an upper layer, it uses them to build a source address and destination address in the header of the PDU. The source address and NS-Quality-of-Service parameters are used to determine which optional functions are to be selected for the network user. At this time, a data unit identifier is assigned to identify this request from other requests. This identifier must remain unique for the lifetime of the initial PDU and any segmented PDUs in the network(s). Subsequent and/or derived PDUs are considered to correspond to the initial PDU if they have the same source address, destination address, and data unit identifier. At first glance, this rule may appear to be connection oriented, but it applies only to PDUs created as a result of segmentation.

The PDU is then forwarded through the subnetwork(s). Each hop examines the destination address to determine if the PDU has reached its destination. If the destination address equals an NSAP (network

service access point) served by the network entity, it has reached its destination. Otherwise, it must be forwarded to the next node.

When the PDU reaches its destination, the receiver removes the protocol control information (PCI) from the PDU. It also uses the addresses in the header to generate the NS-Source-Address and NS-Destination-Address parameters of the UNITDATA indication primitive. It preserves the data field of the PDU until all segments (if any) have been received. The options part of the PDU header is used to invoke any QOS parameters at the receiving end.

Unlike connection-oriented networks, an ISO 8473 connectionless-mode network has more flexibility in terminating service to a user. For example, if a new network connection request is received with a higher priority than an ongoing data transfer, the network may release the lower-priority transfer. Moreover, users may also have priorities established for the data transfers. The network connections (NC) with a higher priority will have their requests serviced first, and the remaining resources of the network are used to attempt to satisfy the lower-priority network connections.

User data is given a specific *lifetime* in the network(s). This mechanism is useful for several reasons. First, it prevents lost or misdirected data from accumulating and consuming network resources. Second, it gives the transmitting entity some control over the disposition of aged data units. Third, it greatly simplifies congestion control, flow control, and accountability logic in the network(s). (It should be emphasized that discarded data units can be recovered by the transport layer.)

The lifetime field in the PDU header is set by the originating network entity. The value also applies to any segmented PDUs and is copied into the header of these data units. The value contains, at any time, the remaining lifetime of the PDU. It is decremented by each network entity that processes it. The value is represented in units of 500 ms and is decremented by one unit by each entity. In the event that delays exceed 500 ms, the value is decremented by more than one unit.

If the lifetime value reaches zero before it reaches its destination, it is discarded. If the error report bit is on, an error report data unit is generated to inform the originator of the lost data. This feature may be considered by some to be a connection-oriented service. Whatever its name, it is obviously very useful if end-to-end accountability is required.

In addition to discarding aged data units, the network may also discard data for other reasons:

- Checksum reveals an error (if checksum is used).

- PDU is received which contains an unsupported function, such as a QOS function.

Local congestion occurs at the receiver.

Header cannot be interpreted accurately.

A PDU cannot be segmented, and it is too large for the underlying network to handle.

The destination address is not reachable or a path route is not acceptable to the network serving the PDU.

The PDU is transferred on a hop-to-hop (next node) basis. The selection of the next system in the route may be influenced by the QOS parameters and other optional parameters. For example, the next hop might be chosen because it supports the QOS requests.

Internetwork routing. The routing technique in ISO 8473 is called source routing and is quite similar to IP source routing. The route is determined by the originator of the PDU by placing a list of the intermediate routes in the options part of the PDU header. The route indicators are called *titles*. The relaying is accomplished by each network entity routing the PDU to the next title in the list. The indicator is updated by each relay point to identify the next stage of the route.

The protocol provides two types of routing: complete source routing and partial source routing. *Complete source routing* requires that the specified route be used in the exact order as in the route list. If this route cannot be taken, the PDU must be discarded with the option of returning an error report to the originator. *Partial source routing* allows a system to route to intermediate systems that are not specified in the list.

Another routing option, called the *record route function*, requires the intermediate systems to add their title to a record route list in the options field of the header. The list is built as the PDU is forwarded to its destination. This list can be used for troubleshooting internet or subnet problems. It can also be used as a tool for efficiency analysis, audit trails, or simply as a "directory" for a returning PDU.

If this option is invoked, either complete route recording or partial route recording is used. With the former, all intermediate systems are recorded and PDU reassembly is performed only if all derived PDUs took the same route. Otherwise, the PDU is discarded. The partial route recording also requires the full list but does not require that the derived PDUs visit the same intermediate systems.

CLNP and IP

The ISO 8473 standard was derived from the IP. From the standpoint of design, the routing should remain the same when an organization

migrates to ISO 8473. The network topology should not be affected by the transition to ISO 8473. So the good news is that the two protocols are similar, and the bad news is that the conversion will still require rewriting a lot of software.

Summary

The IP is a widely used gateway protocol. It is implemented on both LANs and WANs and is designed to operate in a connectionless mode. Notwithstanding, it supports a number of quality service options called type of service (TOS). IP remains transparent to the underlying network. Consequently, it can be placed on a variety of networks. IP does not perform route discovery. It uses routing tables created by route discovery protocols.