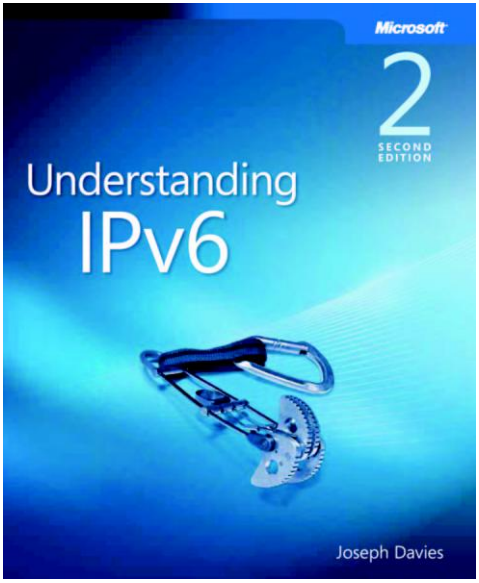

Understanding IPv6, Second Edition

Joseph Davies

To learn more about this book, visit Microsoft Learning at
http://www.microsoft.com/MSPress/books/11607aspx

9780735624467

Microsoft Press

Table of Contents

Foreword . . . . . *xxxi*
Preface . . . . . *xxxiii*
Acknowledgments . . . . . *xxxv*
Introduction . . . . . *xxxvii*
Who Should Read This Book . . . . . *xxxviii*
What You Should Know Before Reading This Book . . . . . *xxxviii*
Organization of This Book . . . . . *xxxix*
Appendices of This Book . . . . . *xxxix*
About the Companion CD-ROM . . . . . *xl*
System Requirements . . . . . *xli*
IPv6 Protocol and Windows Product Versions . . . . . *xli*
A Special Note to Teachers and Instructors . . . . . *xli*
Disclaimers and Support . . . . . *xlii*
Technical Support . . . . . *xlii*

1 Introduction to IPv6 1

Limitations of IPv4 1
Consequences of the Limited IPv4 Address Space 2
Features of IPv6 6
New Header Format 6
Large Address Space 6
Stateless and Stateful Address Configuration 6
IPsec Header Support Required 7
Better Support for Prioritized Delivery 7
New Protocol for Neighboring Node Interaction 7
Extensibility 7
Comparison of IPv4 and IPv6 8
IPv6 Terminology 9

The Case for IPv6 Deployment 11
IPv6 Solves the Address Depletion Problem 12
IPv6 Solves the Disjoint Address Space Problem 12
IPv6 Solves the International Address Allocation Problem 12
IPv6 Restores End-to-End Communication 13
IPv6 Uses Scoped Addresses and Address Selection 13
IPv6 Has More Efficient Forwarding 14
IPv6 Has Support for Security and Mobility 14
Testing for Understanding 15

2 IPv6 Protocol for Windows Server 2008 and Windows Vista 17

Architecture of the IPv6 Protocol for Windows Server 2008 and Windows Vista 17
Features of the IPv6 Protocol for Windows Server 2008 and Windows Vista 19
Installed, Enabled, and Preferred by Default 20
Basic IPv6 Stack Support 20
IPv6 Stack Enhancements 21
GUI and Command-Line Configuration 22
Integrated IPsec Support 22
Windows Firewall Support 22
Temporary Addresses 22
Random Interface IDs 23
DNS Support 23
Source and Destination Address Selection 23
Support for ipv6-literal.net Names 24
LLMNR 24
PNRP 24
Literal IPv6 Addresses in URLs 25
Static Routing 25
IPv6 over PPP 25
DHCPv6 26
ISATAP 26
6to4 26
Teredo 26
PortProxy 27
Application Support 27

Application Programming Interfaces . . . 27
Windows Sockets . . . 28
Winsock Kernel . . . 28
Remote Procedure Call . . . 28
IP Helper . . . 29
Win32 Internet Extensions . . . 29
.NET Framework . . . 29
Windows Filtering Platform . . . 29
Manually Configuring the IPv6 Protocol . . . 30
Configuring IPv6 Through the Properties of Internet Protocol Version 6 (TCP/IPv6) . . . 30
Configuring IPv6 with the Netsh.exe Tool . . . 33
Disabling IPv6 . . . 36
IPv6-Enabled Tools . . . 37
Ipconfig . . . 37
Route . . . 38
Ping . . . 39
Tracert . . . 41
Pathping . . . 42
Netstat . . . 43
Displaying IPv6 Configuration with Netsh . . . 45
Netsh interface ipv6 show interface . . . 45
Netsh interface ipv6 show address . . . 46
Netsh interface ipv6 show route . . . 46
Netsh interface ipv6 show neighbors . . . 47
Netsh interface ipv6 show destinationcache . . . 47
References . . . 47
Testing for Understanding . . . 48

3 IPv6 Addressing . . . 49

The IPv6 Address Space . . . 49
IPv6 Address Syntax . . . 50
Compressing Zeros . . . 51
IPv6 Prefixes . . . 52
Types of IPv6 Addresses . . . 53
Unicast IPv6 Addresses . . . 54
Global Unicast Addresses . . . 54
Topologies Within Global Addresses . . . 55

Local-Use Unicast Addresses 56
Unique Local Addresses 59
Special IPv6 Addresses 60
Transition Addresses 60
Multicast IPv6 Addresses 61
Solicited-Node Address 63
Mapping IPv6 Multicast Addresses to Ethernet Addresses 64
Anycast IPv6 Addresses 65
Subnet-Router Anycast Address 65
IPv6 Addresses for a Host 66
IPv6 Addresses for a Router 66
Subnetting the IPv6 Address Space 67
Step 1: Determining the Number of Subnetting Bits 68
Step 2: Enumerating Subnetted Address Prefixes 69
IPv6 Interface Identifiers 73
EUI-64 Address-Based Interface Identifiers 74
Temporary Address Interface Identifiers 78
IPv4 Addresses and IPv6 Equivalents 79
References 79
Testing for Understanding 80

4 The IPv6 Header 83

Structure of an IPv6 Packet 83
IPv4 Header 84
IPv6 Header 86
Values of the Next Header Field 88
Comparing the IPv4 and IPv6 Headers 89
IPv6 Extension Headers 91
Extension Headers Order 92
Hop-by-Hop Options Header 93
Destination Options Header 97
Routing Header 99
Fragment Header 101
Authentication Header 104
Encapsulating Security Payload Header and Trailer 105
IPv6 MTU 106
Upper-Layer Checksums 107

References 107
Testing for Understanding 108

5 ICMPv6 109

ICMPv6 Overview 109
Types of ICMPv6 Messages 110
ICMPv6 Header 110
ICMPv6 Error Messages 111
Destination Unreachable 111
Packet Too Big 113
Time Exceeded 114
Parameter Problem 115
ICMPv6 Informational Messages 116
Echo Request 116
Echo Reply 117
Comparing ICMPv4 and ICMPv6 Messages 118
Path MTU Discovery 119
Changes in PMTU 120
References 121
Testing for Understanding 121

6 Neighbor Discovery 123

Neighbor Discovery Overview 123
Neighbor Discovery Message Format 125
Neighbor Discovery Options 125
Source and Target Link-Layer Address Options 126
Prefix Information Option 128
Redirected Header Option 130
MTU Option 131
Route Information Option 133
Neighbor Discovery Messages 135
Router Solicitation 136
Router Advertisement 137
Neighbor Solicitation 140
Neighbor Advertisement 142
Redirect 145
Summary of Neighbor Discovery Messages and Options 146

Neighbor Discovery Processes . . . 147
Conceptual Host Data Structures . . . 148
Address Resolution . . . 149
Neighbor Unreachability Detection . . . 152
Duplicate Address Detection . . . 156
Router Discovery . . . 159
Redirect Function . . . 164
Host Sending Algorithm . . . 167
IPv4 Neighbor Messages and Functions and IPv6 Equivalents . . . 169
References . . . 169
Testing for Understanding . . . 169

7 Multicast Listener Discovery and MLD Version 2 . . . 171

MLD and MLDv2 Overview . . . 171
IPv6 Multicast Overview . . . 171
Host Support for Multicast . . . 172
Router Support for Multicast . . . 173
MLD Packet Structure . . . 176
MLD Messages . . . 177
Multicast Listener Query . . . 177
Multicast Listener Report . . . 178
Multicast Listener Done . . . 180
Summary of MLD . . . 182
MLDv2 Packet Structure . . . 182
MLDv2 Messages . . . 182
The Modified Multicast Listener Query . . . 182
MLDv2 Multicast Listener Report . . . 184
Summary of MLDv2 . . . 188
MLD and MLDv2 Support in Windows Server 2008 and Windows Vista . . . 188
References . . . 189
Testing for Understanding . . . 189

8 Address Autoconfiguration . . . 191

Address Autoconfiguration Overview . . . 191
Types of Autoconfiguration . . . 191
Autoconfigured Address States . . . 192
Autoconfiguration Process . . . 193

DHCPv6 196
DHCPv6 Messages 197
DHCPv6 Stateful Message Exchange 201
DHCPv6 Stateless Message Exchange 201
DHCPv6 Support in Windows 201
IPv6 Protocol for Windows Server 2008 and Windows Vista Autoconfiguration Specifics 205
Autoconfigured Addresses for the IPv6 Protocol for Windows Server 2008 and Windows Vista 206
References 208
Testing for Understanding 208

9 IPv6 and Name Resolution 209

Name Resolution for IPv6 209
DNS Enhancements for IPv6 209
LLMNR 210
Source and Destination Address Selection 213
Source Address Selection Algorithm 215
Destination Address Selection Algorithm 217
Example of Using Address Selection 219
Name Resolution Support in Windows Server 2008 and Windows Vista 221
Hosts File 222
DNS Resolver 222
DNS Server Service 223
DNS Dynamic Update 224
Source and Destination Address Selection 225
LLMNR Support 226
Support for ipv6-literal.net Names 227
Peer Name Resolution Protocol 228
References 229
Testing for Understanding 230

10 IPv6 Routing 231

Routing in IPv6 231
IPv6 Routing Table Entry Types 232
Route Determination Process 232
Strong and Weak Host Behaviors 233
Example IPv6 Routing Table for Windows Server 2008 and Windows Vista 234

End-to-End IPv6 Delivery Process 238
IPv6 on the Sending Host 238
IPv6 on the Router 239
IPv6 on the Destination Host 241
IPv6 Routing Protocols 245
Overview of Dynamic Routing 245
Routing Protocol Technologies 246
Routing Protocols for IPv6 247
Static Routing with the IPv6 Protocol for Windows Server 2008 and Windows Vista 249
Configuring Static Routing with Netsh 249
Configuring Static Routing with Routing and Remote Access 253
Dead Gateway Detection 254
References 255
Testing for Understanding 256

11 IPv6 Transition Technologies 259

Overview 259
Node Types 260
IPv6 Transition Addresses 260
Transition Mechanisms 262
Using Both IPv4 and IPv6 262
IPv6-over-IPv4 Tunneling 264
DNS Infrastructure 266
Tunneling Configurations 267
Router-to-Router 267
Host-to-Router and Router-to-Host 268
Host-to-Host 269
Types of Tunnels 270
PortProxy 271
References 273
Testing for Understanding 274

12 ISATAP 275

ISATAP Overview 275
ISATAP Tunneling 276
ISATAP Tunneling Example 277
ISATAP Components 279

Router Discovery for ISATAP Hosts 280
Resolving the Name "ISATAP" 281
Using the netsh interface isatap set router Command 285
ISATAP Addressing Example 285
ISATAP Routing 286
ISATAP Communication Examples 287
ISATAP Host to ISATAP Host 287
ISATAP Host to IPv6 Host 288
Configuring an ISATAP Router 290
References 292
Testing for Understanding 292

13 6to4 295

6to4 Overview 295
6to4 Tunneling 296
6to4 Tunneling Example 297
6to4 Components 298
6to4 Addressing Example 300
6to4 Routing 301
6to4 Support in Windows Server 2008 and Windows Vista 302
6to4 Host/Router Support 302
6to4 Router Support 303
6to4 Communication Examples 306
6to4 Host to 6to4 Host/Router 306
6to4 Host to IPv6 Host 308
Example of Using ISATAP and 6to4 Together 312
Part 1: From ISATAP Host A to 6to4 Router A 314
Part 2: From 6to4 Router A to 6to4 Router B 315
Part 3: From 6to4 Router B to ISATAP Host B 315
References 316
Testing for Understanding 316

14 Teredo 317

Introduction to Teredo 317
Benefits of Using Teredo 318
Teredo Support in Microsoft Windows 318
Teredo and Protection from Unsolicited Incoming IPv6 Traffic 319
Network Address Translators (NATs) 319

Teredo Components . . . 321
Teredo Client . . . 321
Teredo Server . . . 322
Teredo Relay . . . 323
Teredo Host-Specific Relay . . . 323
The Teredo Client and Host-Specific Relay in Windows . . . 324
Teredo Addresses . . . 325
Teredo Packet Formats . . . 329
Teredo Data Packet Format . . . 329
Teredo Bubble Packets . . . 329
Teredo Indicators . . . 330
Teredo Routing . . . 332
Routing for the Teredo Client in Windows . . . 333
Teredo Processes . . . 334
Initial Configuration for Teredo Clients . . . 335
Maintaining the NAT Mapping . . . 339
Initial Communication Between Teredo Clients on the Same Link . . . 339
Initial Communication Between Teredo Clients in Different Sites . . . 340
Initial Communication from a Teredo Client to a Teredo Host-Specific Relay . . . 343
Initial Communication from a Teredo Host-Specific Relay to a Teredo Client . . . 345
Initial Communication from a Teredo Client to an IPv6-Only Host . . . 347
Initial Communication from an IPv6-Only Host to a Teredo Client . . . 350
References . . . 353
Testing for Understanding . . . 353

15 IPv6 Security Considerations . . . 355

IPv6 Security Considerations . . . 355
Authorization for Automatically Assigned Addresses and Configurations . . . 355
Recommendations . . . 356
Protection of IPv6 Packets . . . 356
Recommendations . . . 357
Host Protection from Scanning and Attacks . . . 357
Address Scanning . . . 357
Port Scanning . . . 358
Recommendations . . . 358

Control of What Traffic Is Exchanged with the Internet 358
Recommendations 359
Summary 360
References 360
Testing for Understanding 360

16 Deploying IPv6 363

Introduction 363
Planning for IPv6 Deployment 363
Platform Support for IPv6 364
Application Support for IPv6 364
Unicast IPv6 Addressing 365
Tunnel-Based IPv6 Connectivity 366
Native IPv6 Connectivity 369
Name Resolution with DNS 370
DHCPv6 370
Host-Based Security and IPv6 Traffic 371
Prioritized Delivery for IPv6 Traffic 371
Deploying IPv6 372
Set Up an IPv6 Test Network 373
Begin Application Migration 373
Configure DNS Infrastructure to Support AAAA Records and Dynamic Updates 375
Deploy a Tunneled IPv6 Infrastructure with ISATAP 375
Upgrade IPv4-Only Hosts to IPv6/IPv4 Hosts 376
Begin Deploying a Native IPv6 Infrastructure 376
Connect Portions of Your Intranet over the IPv4 Internet 378
Connect Portions of Your Intranet over the IPv6 Internet 379
Summary 379
References 380
Testing for Understanding 380

A Link-Layer Support for IPv6 381

Basic Structure of IPv6 Packets 381
LAN Media 381
Ethernet: Ethernet II 382
Ethernet: IEEE 802.3 SNAP 383
Token Ring: IEEE 802.5 SNAP 385
FDDI 386

IEEE 802.11. 388
WAN Media 391
PPP 392
X.25 393
Frame Relay 395
ATM: Null Encapsulation 396
ATM: SNAP Encapsulation 398
IPv6 over IPv4 399
References 399

B Windows Sockets Changes for IPv6 401

Added Constants 401
Address Data Structures. 402
in6_addr 402
sockaddr_in6 402
sockaddr_storage 403
Wildcard Addresses 403
in6addr_loopback and IN6ADDR_LOOPBACK_INIT 403
Core Sockets Functions 404
Name-to-Address Translation. 404
Address-to-Name Translation. 406
Using *getaddrinfo* 407
Address Conversion Functions 407
Socket Options 407
New Macros. 408
References 409

C IPv6 RFC Index. 411

General 411
Addressing 411
Applications. 412
Sockets API 412
Transport Layer 412
Internet Layer 413
Network Layer Security 414
Link Layer. 414
Routing. 415
IPv6 Transition Technologies. 415

D Testing for Understanding Answers 417

Chapter 1: Introduction to IPv6 417
Chapter 2: IPv6 Protocol for Windows Server 2008 and Windows Vista 418
Chapter 3: IPv6 Addressing 420
Chapter 4: The IPv6 Header 423
Chapter 5: ICMPv6 424
Chapter 6: Neighbor Discovery 425
Chapter 7: Multicast Listener Discovery and MLD Version 2 428
Chapter 8: Address Autoconfiguration 430
Chapter 9: IPv6 and Name Resolution 431
Chapter 10: IPv6 Routing 432
Chapter 11: IPv6 Transition Technologies 434
Chapter 12: ISATAP 435
Chapter 13: 6to4 436
Chapter 14: Teredo 437
Chapter 15: IPv6 Security Considerations 438
Chapter 16: Deploying IPv6 439

E Setting Up an IPv6 Test Lab 441

IPv6 Test Lab Setup 441
DNS1 443
CLIENT1 443
ROUTER1 444
ROUTER2 444
CLIENT2 445
IPv6 Test Lab Tasks 446
Performing Link-Local Pings 446
Enabling Native IPv6 Connectivity on Subnet 1 447
Configuring ISATAP 448
Configuring Native IPv6 Connectivity for All Subnets 449
Using Name Resolution 451
Configuring an IPv6-Only Routing Infrastructure 452

F Mobile IPv6 453

Overview 453
Mobile IPv6 Components 453
Mobile IPv6 Transport Layer Transparency 455

Mobile IPv6 Messages and Options 456
Mobility Header and Messages 456
Type 2 Routing Header 458
Home Address Option for the Destination Options Header 459
ICMPv6 Messages for Mobile IPv6 460
Modifications to Neighbor Discovery Messages and Options 462
Mobile IPv6 Data Structures 465
Binding Cache 465
Binding Update List 466
Home Agents List 467
Correspondent Registration 468
Return Routability Procedure 469
Detecting Correspondent Nodes That Are Not Mobile IPv6–Capable 471
Mobile IPv6 Message Exchanges 471
Data Between a Mobile Node and a Correspondent Node 471
Binding Maintenance 478
Home Agent Discovery 483
Mobile Prefix Discovery 484
Mobile IPv6 Processes 486
Attaching to the Home Link 487
Moving from the Home Link to a Foreign Link 488
Moving to a New Foreign Link 496
Returning Home 499
Mobile IPv6 Host Sending Algorithm 502
Mobile IPv6 Host Receiving Algorithm 505
References 508

G IPv6 Reference Tables 509

Glossary 515

Index 529

Chapter 3

IPv6 Addressing

At the end of this chapter, you should be able to do the following:

- Describe the IPv6 address space, and state why the address length of 128 bits was chosen.
- Describe IPv6 address syntax, including zero suppression and compression and prefixes.
- Enumerate and describe the function of the different types of unicast IPv6 addresses.
- Describe the format of multicast IPv6 addresses.
- Describe the function of anycast IPv6 addresses.
- Describe how IPv6 interface identifiers are determined.
- Describe how to perform bit-level subnetting on the subnet identifier portion of a unicast IPv6 address prefix.
- List and compare the different addressing concepts between IPv4 addresses and IPv6 addresses.

The IPv6 Address Space

The most obvious distinguishing feature of Internet Protocol version 6 (IPv6) is its use of much larger addresses. The size of an address in IPv6 is 128 bits, a bit-string that is four times longer than the 32-bit IPv4 address. A 32-bit address space allows for 2^{32}, or 4,294,967,296, possible addresses. A 128-bit address space allows for 2^{128}, or 340,282,366,920,938,463,463, 374,607,431,768,211,456 (3.4×10^{38} or 340 undecillion), possible addresses.

In the late 1970s, when the IPv4 address space was designed, it was unimaginable that it could ever be exhausted. However, the administrative procedures that defined address allocation did not anticipate the recent explosion of hosts on the Internet. The IPv4 address space was thus consumed to the point that, by 1992, it was clear a replacement would be necessary.

With IPv6, it is even more difficult to conceive that the IPv6 address space will ever be consumed. To help put this number in perspective, a 128-bit address space provides 6.65×10^{23} addresses for each square meter of the Earth's surface.

It is important to remember that the decision to make the IPv6 address 128 bits in length was not so that every square meter of the Earth could have 6.65×10^{23} addresses. Rather, the relatively large size of the IPv6 address is designed to be divided into hierarchical unicast routing domains that reflect the topology of the modern-day Internet. The use of 128 bits allows for multiple levels of hierarchy and flexibility in designing hierarchical unicast addressing and routing that is currently lacking on the IPv4-based Internet.

It is easy to get lost in the vastness of the IPv6 address space. As we will discover, the unthinkably large 128-bit IPv6 address that is assigned to an interface on a typical IPv6 host is composed of a 64-bit subnet prefix and a 64-bit interface identifier (a 50-50 split between subnet space and interface space). The 64 bits of subnet prefix leave enough addressing room to satisfy the addressing requirements of the levels of Internet service providers (ISPs) between your organization and the backbone of the Internet and the addressing needs of your organization. The 64 bits of interface identifier accommodate the mapping of current and future link-layer media access control (MAC) addresses.

IPv6 Address Syntax

IPv4 addresses are represented in dotted-decimal format. The 32-bit IPv4 address is divided along 8-bit boundaries. Each set of 8 bits is converted to its decimal equivalent and separated by periods. For IPv6, the 128-bit address is divided along 16-bit boundaries, and each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is called *colon hexadecimal*.

The following is an IPv6 address in binary form:

```
0010000000000001000011011011100000000000000000000010111100111011
0000001010101010000000001111111111111111000101000100111000101101O
```

The 128-bit address is divided along 16-bit boundaries:

```
0010000000000001    0000110110111000    0000000000000000    0010111100111011
0000001010101010    0000000011111111    1111111000101000    1001110001011010
```

Each 16-bit block is converted to hexadecimal and delimited with colons. The result is the following:

```
2001:0DB8:0000:2F3B:02AA:00FF:FE28:9C5A
```

IPv6 address representation is further simplified by suppressing the leading zeros within each 16-bit block. However, each block must have at least a single digit. With leading zero suppression, the result is the following:

```
2001:DB8:0:2F3B:2AA:FF:FE28:9C5A
```

Number System Choice for IPv6

IPv6 uses hexadecimal (the $Base_{16}$ numbering system), rather than decimal (the $Base_{10}$ numbering system), because it is easier to convert between hexadecimal and binary than it is to convert between decimal and binary. Each hexadecimal digit represents four binary digits.

With IPv4, decimal is used to make the IPv4 addresses more palatable for humans and a 32-bit address becomes 4 decimal numbers separated by the period (.) character. With IPv6, dotted-decimal representation would result in 16 decimal numbers separated by the period (.) character. IPv6 addresses are so large that there is no attempt to make them palatable to most humans. Configuration of typical end systems is automated, and end users will almost always use names rather than IPv6 addresses. Therefore, the addresses are expressed in a way to make them more palatable to computers and IPv6 network administrators who understand the semantics and relationship of hexadecimal and binary numbers.

Table 3-1 lists the conversion between binary, hexadecimal, and decimal numbers.

Table 3-1 Converting Between Binary, Hexadecimal, and Decimal Numbers

Binary	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Compressing Zeros

Some types of IPv6 addresses contain long sequences of zeros. To further simplify the representation of IPv6 addresses, a single contiguous sequence of 16-bit blocks set to 0 in the colon hexadecimal format can be compressed to ::, known as a *double colon*. For example, the link-local address of FE80:0:0:0:2AA:FF:FE9A:4CA2 can be compressed to FE80::2AA:FF:FE9A:4CA2. The multicast address FF02:0:0:0:0:0:0:2 can be compressed to FF02::2.

Note You cannot use zero compression to include part of a 16-bit block. For example, you cannot express FF02:30:0:0:0:0:0:5 as FF02:3::5, but FF02:30::5 is correct.

How Many Blocks or Bits in ::?

To determine how many 0 blocks are represented by the ::, you can count the number of blocks in the compressed address and subtract this number from 8. To determine how many 0 bits are represented by the ::, multiply the number of blocks the :: represents by 16. For example, in the address FF02::2, there are two blocks (the "FF02" block and the "2" block). The number of blocks expressed by the :: is 6 (8 − 2). The number of bits expressed by the :: is 96 (96 = 6 × 16). Zero compression can be used only once in a given address. Otherwise, you could not determine the number of 0 blocks or bits represented by each instance of ::.

IPv6 Prefixes

The prefix is the part of the address where the bits have fixed values or are the bits that define a route or subnet. Prefixes for IPv6 subnets and summarized routes are expressed in the same way as Classless Inter-Domain Routing (CIDR) notation for IPv4. An IPv6 prefix is written in *address/prefix-length* notation.

For example, 2001:DB8:2A0:2F3B::/64 is a subnet prefix and 2001:DB8:3F::/48 is a summarized route prefix. As described earlier in this chapter, the 64-bit prefix is used for individual subnets to which nodes are attached. All subnets have a 64-bit prefix. Any prefix that is less than 64 bits is a summarized route or an address range that is summarizing a portion of the IPv6 address space.

Note IPv4 implementations commonly use a dotted-decimal representation of the prefix length known as the *subnet mask*. A subnet mask is not used for IPv6. Only the prefix length notation is supported.

An IPv6 prefix is relevant only for routes or address ranges, not for individual unicast addresses. In IPv4, it is common to express an IPv4 address with its prefix length. For example, 192.168.29.7/24 (equivalent to 192.168.29.7 with the subnet mask 255.255.255.0) denotes the IPv4 address 192.168.29.7 with a 24-bit subnet mask. Because IPv4 addresses are no longer class-based, you cannot assume the class-based subnet mask based on the value of the leading octet. The prefix length is included so that you can determine which bits identify the subnet and which bits identify the host on the subnet. Because the number of bits used to identify the subnet in IPv4 is variable, the prefix length is needed to separate the subnet prefix from the host ID.

In common IPv6 practice, however, there is no notion of a variable-length subnet prefix. At the individual IPv6 subnet level for currently defined unicast IPv6 addresses, the number of bits used to identify the subnet is always 64 and the number of bits used to identify the host on the subnet is always 64. Therefore, while unicast IPv6 addresses written with their prefix lengths

are permitted in RFC 4291, in practice their prefix lengths are always 64 and therefore do not need to be expressed. For example, there is no need to express the IPv6 unicast address 2001:DB8::2AC4:2AA:FF:FE9A:82D4 as 2001:DB8::2AC4:2AA:FF:FE9A:82D4/64. Because of the 50-50 split of subnet prefixes and interface identifiers, the unicast IPv6 address 2001:DB8::2AC4:2AA:FF:FE9A:82D4 implies that the subnet prefix is 2001:DB8:0:0:2AC4::/64.

Note Address prefixes with a prefix length longer than 64 bits can be used for point-to-point links between routers.

Types of IPv6 Addresses

There are three types of IPv6 addresses:

Unicast

A unicast address identifies a single interface within the scope of the type of address. The scope of an address is the region of the IPv6 network over which the address is unique. With the appropriate unicast routing topology, packets addressed to a unicast address are delivered to a single interface. To accommodate load-balancing systems, RFC 4291 allows for multiple interfaces to use the same address as long as they appear as a single interface to the IPv6 implementation on the host.

Multicast

A multicast address identifies zero or more interfaces on the same or different hosts. With the appropriate multicast routing topology, packets addressed to a multicast address are delivered to all interfaces identified by the address.

Anycast

An anycast address identifies multiple interfaces. With the appropriate unicast routing topology, packets addressed to an anycast address are delivered to a single interface—the nearest interface that is identified by the address. The nearest interface is defined as being the closest in terms of routing distance. A multicast address is used for one-to-many communication, with delivery to multiple interfaces. An anycast address is used for one-to-one-of-many communication, with delivery to a single interface.

In all cases, IPv6 addresses identify interfaces, not nodes. A node is identified by any unicast address assigned to any one of its interfaces.

Note RFC 4291 does not define a broadcast address. All types of IPv4 broadcast addressing are performed in IPv6 using multicast addresses. For example, the subnet and limited broadcast addresses from IPv4 are replaced with the link-local scope all-nodes multicast address of FF02::1.

Unicast IPv6 Addresses

The following types of addresses are unicast IPv6 addresses:

- Global unicast addresses
- Link-local addresses
- Site-local addresses
- Unique local addresses
- Special addresses
- Transition addresses

Global Unicast Addresses

IPv6 global addresses are equivalent to public IPv4 addresses. They are globally routable and reachable on the IPv6 Internet. Global unicast addresses are designed to be aggregated or summarized for an efficient routing infrastructure. Unlike the current IPv4-based Internet, which is a mixture of both flat and hierarchical routing, the IPv6-based Internet has been designed from its foundation to support efficient, hierarchical addressing and routing. The scope of a global address is the entire IPv6 Internet.

RFC 4291 defines global addresses as all addresses that are not the unspecified, loopback, link-local unicast, or multicast addresses (described later in this chapter). However, Figure 3-1 shows the structure of global unicast addresses defined in RFC 3587 that are currently being used on the IPv6 Internet.

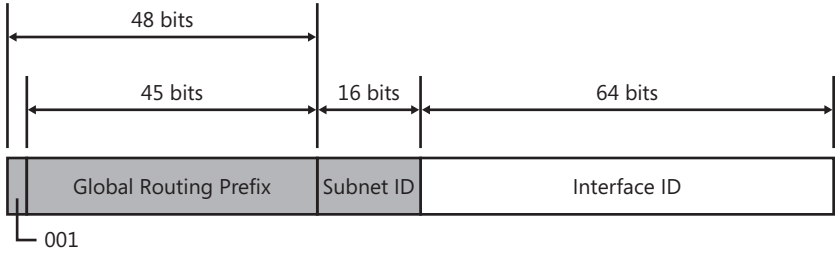

Figure 3-1 The structure of global unicast addresses defined in RFC 3587

The fields in the global unicast address are described in the following list:

- **Fixed portion set to 001** The three high-order bits are set to 001.
- **Global Routing Prefix** Indicates the global routing prefix for a specific organization's site. The combination of the three fixed bits and the 45-bit Global Routing Prefix is used to create a 48-bit site prefix, which is assigned to an individual site of an organization. A site is an autonomously operating IP-based network that is connected to the IPv6 Internet. Network architects and administrators within the site determine the addressing plan and

routing policy for the organization network. Once assigned, routers on the IPv6 Internet forward IPv6 traffic matching the 48-bit prefix to the routers of the organization's site.

- **Subnet ID** The Subnet ID is used within an organization's site to identify subnets within its site. The size of this field is 16 bits. The organization's site can use these 16 bits within its site to create 65,536 subnets or multiple levels of addressing hierarchy and an efficient routing infrastructure. With 16 bits of subnetting flexibility, a global unicast prefix assigned to an organization site is equivalent to a public IPv4 Class A address prefix (assuming that the last octet is used for identifying nodes on subnets). The routing structure of the organization's network is not visible to the ISP.
- **Interface ID** Indicates the interface on a specific subnet within the site. The size of this field is 64 bits. The interface ID in IPv6 is equivalent to the node ID or host ID in IPv4.

Trillions of Sites

Another way to gauge the practical size of the IPv6 address space is to examine the number of sites that can connect to the IPv6 Internet. With the current allocation practice defined in RFC 3587 of 48-bit global address prefixes, it is possible to define 2^{45} or 35,184,372,088,832 possible 48-bit prefixes to assign to sites connected to the IPv6 Internet. There are more IPv6 sites than possible IPv4 addresses. This large number of sites is possible even when we are using only one-eighth of the entire IPv6 address space.

By comparison, using the Internet address classes originally defined for IPv4, it was possible to assign 2,113,389 address prefixes to organizations connected to the Internet. The number 2,113,389 is derived from adding up all the possible Class A, Class B, and Class C address prefixes and then subtracting the prefixes used for the private address space. Even with the adoption of CIDR to make more efficient use of unassigned Class A and Class B address prefixes, the number of possible sites connected to the Internet is not substantially increased, nor does it approach the number of possible sites that can be connected to the IPv6 Internet.

Topologies Within Global Addresses

The fields within the global address create a three-level topological structure, as shown in Figure 3-2.

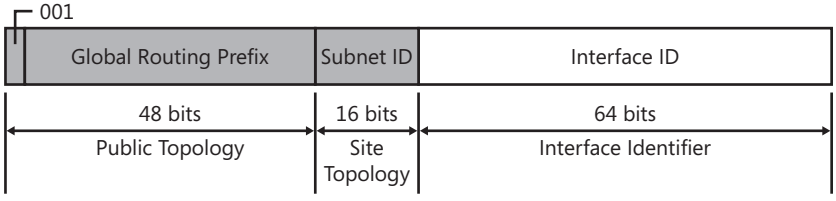

Figure 3-2 The topological structure of the global address

The public topology is the collection of larger and smaller ISPs that provide access to the IPv6 Internet. The site topology is the collection of subnets within an organization's site. The interface identifier specifies a unique interface on a subnet within an organization's site.

Local-Use Unicast Addresses

Local-use unicast addresses do not have a global scope and can be reused. There are two types of local-use unicast addresses:

1. Link-local addresses are used between on-link neighbors and for Neighbor Discovery processes.
2. Site-local addresses are used between nodes communicating with other nodes in the same organization.

Link-Local Addresses

IPv6 link-local addresses, identified by the initial 10 bits being set to 1111 1110 10 and the next 54 bits set to 0, are used by nodes when communicating with neighboring nodes on the same link. For example, on a single-link IPv6 network with no router, link-local addresses are used to communicate between hosts on the link. IPv6 link-local addresses are similar to IPv4 link-local addresses defined in RFC 3927 that use the 169.254.0.0/16 prefix. The use of IPv4 link-local addresses is known as Automatic Private IP Addressing (APIPA) in Windows Vista, Windows Server 2008, Windows Server 2003, and Windows XP. The scope of a link-local address is the local link.

Figure 3-3 shows the structure of the link-local address.

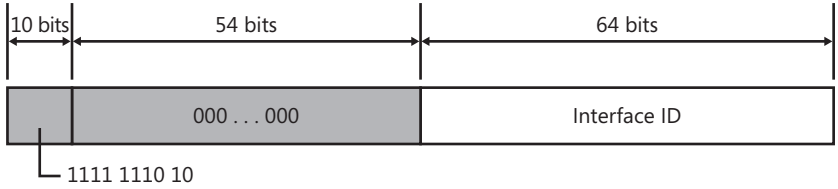

Figure 3-3 The structure of the link-local address

A link-local address is required for some Neighbor Discovery processes and is always automatically configured, even in the absence of all other unicast addresses. For more information about the address autoconfiguration process for link-local addresses, see Chapter 8, "Address Autoconfiguration."

Link-local addresses always begin with FE80. With the 64-bit interface identifier, the prefix for link-local addresses is always FE80::/64. An IPv6 router never forwards link-local traffic beyond the link.

Site-Local Addresses

Site-local addresses, identified by setting the first 10 bits to 1111 1110 11, are equivalent to the IPv4 private address space (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16). For example, private intranets that do not have a direct, routed connection to the IPv6 Internet can use site-local addresses without conflicting with global addresses. Site-local addresses are not reachable from other sites, and routers must not forward site-local traffic outside the site. Site-local addresses can be used in addition to global addresses. The scope of a site-local address is the site.

Figure 3-4 shows the structure of the site-local address.

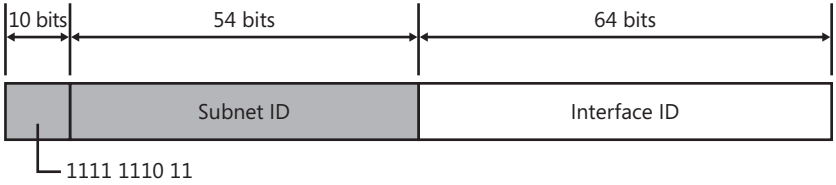

Figure 3-4 The structure of the site-local address

Unlike link-local addresses, site-local addresses are not automatically configured and must be assigned either through stateless or stateful address autoconfiguration. For more information, see Chapter 8.

The first 10 bits are always fixed for site-local addresses, beginning with FEC0::/10. After the 10 fixed bits is a 54-bit Subnet ID field that provides 54 bits with which you can create subnets within your organization. You can have a flat subnet structure, or you can divide the high-order bits of the Subnet ID field to create a hierarchical and summarizable routing infrastructure. After the Subnet ID field is a 64-bit Interface ID field that identifies a specific interface on a subnet.

Site-local addresses have been formally deprecated in RFC 3879 for future IPv6 implementations. However, existing implementations of IPv6 can continue to use site-local addresses.

Zone IDs for Local-Use Addresses

Unlike global addresses, local-use addresses (link-local and site-local addresses) can be reused. Link-local addresses are reused on each link. Site-local addresses can be reused within each site of an organization. Because of this address reuse capability, link-local and site-local addresses are ambiguous. To specify the link on which the destination is located or the site within which the destination is located, an additional identifier is needed. This additional identifier is a zone identifier (ID), also known as a scope ID, which identifies a connected portion of a network that has a specified scope.

The syntax specified in RFC 4007 for identifying the zone associated with a local-use address is *Address*%*zone_ID*, in which *Address* is a local-use unicast IPv6 address and *zone_ID* is an integer value representing the zone. The values of the zone ID are defined relative to the sending host. Therefore, different hosts might determine different zone ID values for the same physical zone. For example, Host A might choose 3 to represent the zone of an attached link and Host B might choose 4 to represent the same link.

For Windows-based IPv6 hosts, the zone IDs for link-local and site-local addresses are defined as follows:

- For link-local addresses, the zone ID is typically the interface index of the interface either assigned the address or to be used as the sending interface for a link-local destination. The interface index is an integer starting at 1 that is assigned to IPv6 interfaces, which include a loopback and one or multiple LAN or tunnel interfaces. Multiple interfaces can have the same link-local zone ID if they are attached to the same link. You can view the list of interface indexes from the display of the **netsh interface ipv6 show interface** command. You must include a zone ID with a link-local destination.
- For site-local addresses, the zone ID is the site ID, an integer assigned to the site of an organization. For organizations that do not reuse the site-local address prefix, the site ID is set to 1 by default and does not need to be specified. In Windows, you can view the site ID from the display of the **netsh interface ipv6 show address level=verbose** command.

The following are examples of using Windows tools and the zone ID:

- **ping fe80::2b0:d0ff:fee9:4143%3** In this case, *3* is the interface index of the interface attached to the link containing the destination address.
- **tracert fec0::f282:2b0:d0ff:fee9:4143%2** In this case, *2* is the site ID of the organization site containing the destination address.

In Windows Vista and Windows Server 2008, the Ipconfig.exe tool displays the zone ID of local-use IPv6 addresses. The following is an excerpt from the display of the **ipconfig** command:

```
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : ecoast.example.com
   IPv6 Address. . . . . . . . . . . : 2001:db8:21da:7:713e:a426:d167:37ab
   Temporary IPv6 Address. . . . . . : 2001:db8:21da:7:5099:ba54:9881:2e54
   Link-local IPv6 Address . . . . . : fe80::713e:a426:d167:37ab%6
   IPv4 Address. . . . . . . . . . . : 157.60.14.11
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::20a:42ff:feb0:5400%6
                                       157.60.14.1
```

For the link-local addresses in the display of the **ipconfig** command, the zone ID indicates the interface index of the interface either assigned the address (for Link-Local IPv6 Address) or the interface through which an address is reachable (for Default Gateway).

Unique Local Addresses

Site-local addresses provide a private addressing alternative to global addresses for intranet traffic. However, because the site-local address prefix can be reused to address multiple sites within an organization, a site-local address prefix can be duplicated. The ambiguity of site-local addresses in an organization adds complexity and difficulty for applications, routers, and network managers. For more information, see section 2 of RFC 3879.

To replace site-local addresses with a new type of address that is private to an organization yet unique across all the sites of the organization, RFC 4193 defines unique local IPv6 unicast addresses. Figure 3-5 shows the structure of the unique local address.

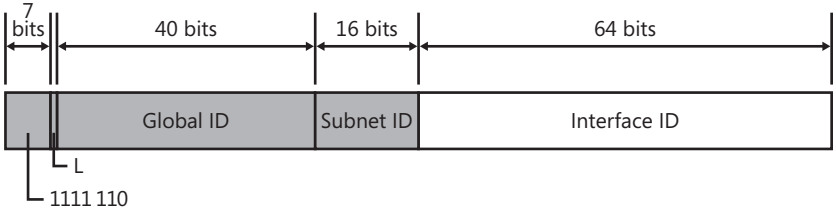

Figure 3-5 The structure of the unique local address

The first 7 bits have the fixed binary value of 1111110. All local addresses have the address prefix FC00::/7. The Local (L) flag is set 1 to indicate that the prefix is locally assigned. The L flag value set to 0 is not defined in RFC 3879. Therefore, unique local addresses within an organization with the L flag set to 1 have the address prefix of FD00::/8. The Global ID identifies a specific site within an organization and is set to a randomly derived 40-bit value. By deriving a random value for the Global ID, an organization can have statistically unique 48-bit prefixes assigned to their sites. Additionally, two organizations that use unique local addresses that merge have a low probability of duplicating a 48-bit unique local address prefix, minimizing site renumbering. Unlike the Global Routing Prefix in global addresses, the Global IDs in unique local address prefixes are not designed to be summarized.

Unique local addresses have a global scope, but their reachability is defined by routing topology and filtering policies at Internet boundaries. Organizations will not advertise their unique local address prefixes outside of their organizations or create DNS entries with unique local addresses in the Internet DNS. Organizations can easily create filtering policies at their Internet boundaries to prevent all unique local-addressed traffic from being forwarded. Because they have a global scope, unique local addresses do not need a zone ID.

The global address and unique local address share the same structure beyond the first 48 bits of the address. In both addresses, the 16-bit Subnet ID field identifies a subnet within an organization. Because of this, you can create a subnetted routing infrastructure that is used for both local and global addresses.

For example, a specific subnet of your organization can be assigned both the global prefix 2001:DB8:4D1C:221A::/64 and the local prefix FD0E:2D:BA9:221A::/64, where the subnet is identified for both types of prefixes by the Subnet ID value of *221A*. Although the subnet identifier is the same for both prefixes, routes for both prefixes must still be propagated throughout the routing infrastructure so that addresses based on both prefixes are reachable.

Special IPv6 Addresses

The following are special IPv6 addresses:

- **Unspecified address**

  The unspecified address (0:0:0:0:0:0:0:0 or ::) is used only to indicate the absence of an address. It is equivalent to the IPv4 unspecified address of 0.0.0.0. The unspecified address is typically used as a source address when a unique address has not yet been determined. The unspecified address is never assigned to an interface or used as a destination address.

- **Loopback address**

  The loopback address (0:0:0:0:0:0:0:1 or ::1) is assigned to a loopback interface, enabling a node to send packets to itself. It is equivalent to the IPv4 loopback address of 127.0.0.1. Packets addressed to the loopback address must never be sent on a link or forwarded by an IPv6 router.

Transition Addresses

To aid in the transition from IPv4 to IPv6 and the coexistence of both types of hosts, the following addresses are defined:

- **IPv4-compatible address**

  The IPv4-compatible address, 0:0:0:0:0:0:*w.x.y.z* or ::*w.x.y.z* (where *w.x.y.z* is the dotted-decimal representation of a public IPv4 address), is used by IPv6/IPv4 nodes that are communicating with IPv6 over an IPv4 infrastructure that uses public IPv4 addresses, such as the Internet. IPv4-compatible addresses are deprecated in RFC 4291 and are not supported in IPv6 for Windows Vista and Windows Server 2008.

- **IPv4-mapped address**

  The IPv4-mapped address, 0:0:0:0:0:FFFF:*w.x.y.z* or ::FFFF: *w.x.y.z*, is used to represent an IPv4 address as a 128-bit IPv6 address.

- **6to4 address**

  An address of the type 2002:*WWXX*:*YYZZ*:*Subnet ID*:*Interface ID*, where *WWXX*:*YYZZ* is the colon hexadecimal representation of *w.x.y.z* (a public IPv4 address), is assigned a node for the 6to4 IPv6 transition technology.

- **ISATAP address**

  An address of the type *64-bit prefix*:0:5EFE:*w.x.y.z*, where *w.x.y.z* is a private IPv4 address, is assigned to a node for the Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) IPv6 transition technology.

- **Teredo address**

  A global address that uses the prefix 2001::/32 and is assigned to a node for the Teredo IPv6 transition technology. Beyond the first 32 bits, Teredo addresses are used to encode the IPv4 address of a Teredo server, flags, and an obscured version of a Teredo client's external address and UDP port number.

For more information about these addresses, see Chapter 11, "IPv6 Transition Technologies."

Multicast IPv6 Addresses

In IPv6, multicast traffic operates in the same way that it does in IPv4. Arbitrarily located IPv6 nodes can listen for multicast traffic on an arbitrary IPv6 multicast address. IPv6 nodes can listen to multiple multicast addresses at the same time. Nodes can join or leave a multicast group at any time.

IPv6 multicast addresses have the first 8 bits set to 1111 1111. Therefore, an IPv6 multicast address always begins with FF. Multicast addresses cannot be used as source addresses or as intermediate destinations in a Routing extension header. Beyond the first 8 bits, multicast addresses include additional structure to identify flags, their scope, and the multicast group. Figure 3-6 shows the structure of the IPv6 multicast address.

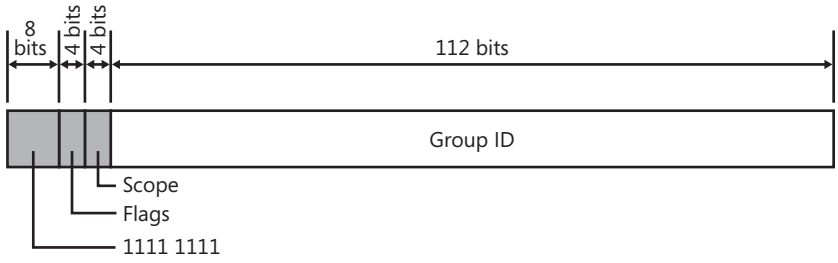

Figure 3-6 The structure of the IPv6 multicast address

The following list describes the fields in the multicast address:

- **Flags** Indicates flags set on the multicast address. The size of this field is 4 bits, consisting of three flags in the low-order bits. The first low-order bit is the Transient (T) flag. When set to 0, the T flag indicates that the multicast address is a permanently assigned (well-known) multicast address allocated by the Internet Assigned Numbers Authority (IANA). When set to 1, the T flag indicates that the multicast address is a transient (not permanently assigned) multicast address. The second low-order bit is for the Prefix (P) flag, which indicates whether the multicast address is based on a unicast address prefix.

RFC 3306 describes the P flag. The third low-order bit is for the Rendezvous Point Address (R) flag, which indicates whether the multicast address contains an embedded rendezvous point address. RFC 3956 describes the R flag.

- **Scope** Indicates the scope of the IPv6 network for which the multicast traffic is intended to be delivered. The size of this field is 4 bits. In addition to using information provided by multicast routing protocols, routers use the multicast scope to determine whether multicast traffic can be forwarded.

Table 3-2 lists the values for the Scope field assigned in RFC 4291. All other values are unassigned.

Table 3-2 Defined Values for the Scope Field

Scope Field Value	Scope
0	Reserved
1	Interface-local scope
2	Link-local scope
3	Reserved
4	Admin-local scope
5	Site-local scope
8	Organization-local scope
E	Global scope
F	Reserved

For example, traffic with the multicast address of FF02::2 has a link-local scope. An IPv6 router never forwards this traffic beyond the local link.

- **Group ID** Identifies the multicast group, and is unique within the scope. The size of this field is 112 bits. Permanently assigned group IDs are independent of the scope. Transient group IDs are relevant only to a specific scope. Multicast addresses from FF01:: through FF0F:: are reserved, well-known addresses.

To identify all nodes for the interface-local and link-local scopes, the following addresses are defined:

- FF01::1 (interface-local scope all-nodes multicast address)
- FF02::1 (link-local scope all-nodes multicast address)

To identify all routers for the interface-local, link-local, and site-local scopes, the following addresses are defined:

- FF01::2 (interface-local scope all-routers multicast address)
- FF02::2 (link-local scope all-routers multicast address)
- FF05::2 (site-local scope all-routers multicast address)

For the current list of permanently assigned IPv6 multicast addresses, see *http://www.iana.org/assignments/ipv6-multicast-addresses*.

IPv6 multicast addresses replace all forms of IPv4 broadcast addresses. The IPv4 network broadcast (in which all host bits are set to 1 in a classful environment), subnet broadcast (in which all host bits are set to 1 in a non-classful environment), and limited broadcast (255.255.255.255) addresses are replaced by the link-local scope all-nodes multicast address (FF02:01) in IPv6.

Solicited-Node Address

The solicited-node address facilitates the efficient querying of network nodes during link-layer address resolution—the resolving of a link-layer address of a known IPv6 address. In IPv4, the Address Resolution Protocol (ARP) Request frame is sent to the MAC-level broadcast, disturbing all nodes on the network segment, including those that are not running IPv4. IPv6 uses the Neighbor Solicitation message to perform link-layer address resolution. However, instead of using the local-link scope all-nodes multicast address as the Neighbor Solicitation message destination, which would disturb all IPv6 nodes on the local link, the solicited-node multicast address is used. The solicited-node multicast address is constructed from the prefix FF02::1:FF00:0/104 and the last 24 bits (6 hexadecimal digits) of a unicast IPv6 address. Figure 3-7 shows the mapping of a unicast IPv6 address and its corresponding solicited-node multicast address.

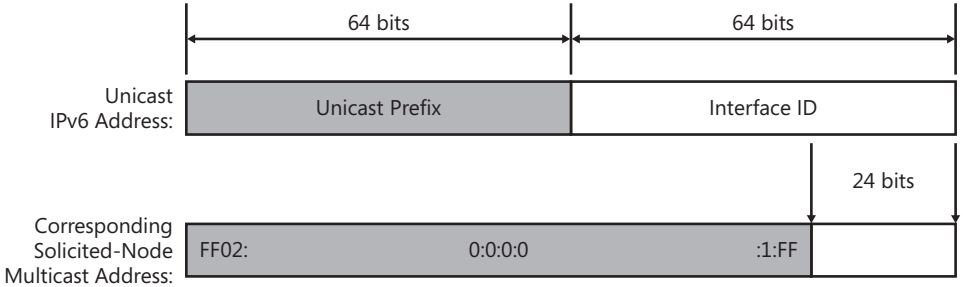

Figure 3-7 The mapping of a unicast address to its solicited-node multicast address

For example, Node A is assigned the link-local address of FE80::2AA:FF:FE28:9C5A and is also listening on the corresponding solicited-node multicast address of FF02::1:FF28:9C5A. (An underline is used to highlight the correspondence of the last six hexadecimal digits.) Node B on the local link must resolve Node A's link-local address FE80::2AA:FF:FE28:9C5A to its corresponding link-layer address. Node B sends a Neighbor Solicitation message to the solicited-node multicast address of FF02::1:FF28:9C5A. Because Node A is listening on this multicast address, it processes the Neighbor Solicitation message and sends a unicast Neighbor Advertisement message in reply.

The result of using the solicited-node multicast address is that link-layer address resolutions, a common occurrence on a link, are not using a mechanism that disturbs all network nodes. By using the solicited-node address, very few nodes are disturbed during address resolution. In practice, because of the relationship between the IPv6 interface ID and the solicited-node address, the solicited-node address acts as a pseudo-unicast address for very efficient address resolution. For more information, see the "IPv6 Interface Identifiers" section in this chapter.

Mapping IPv6 Multicast Addresses to Ethernet Addresses

When sending IPv6 multicast packets on an Ethernet link, the corresponding destination MAC address is 0x33-33-mm-mm-mm-mm, where *mm-mm-mm-mm* is a direct mapping of the last 32 bits (8 hexadecimal digits) of the IPv6 multicast address. Figure 3-8 shows the mapping of an IPv6 multicast address to an Ethernet multicast address.

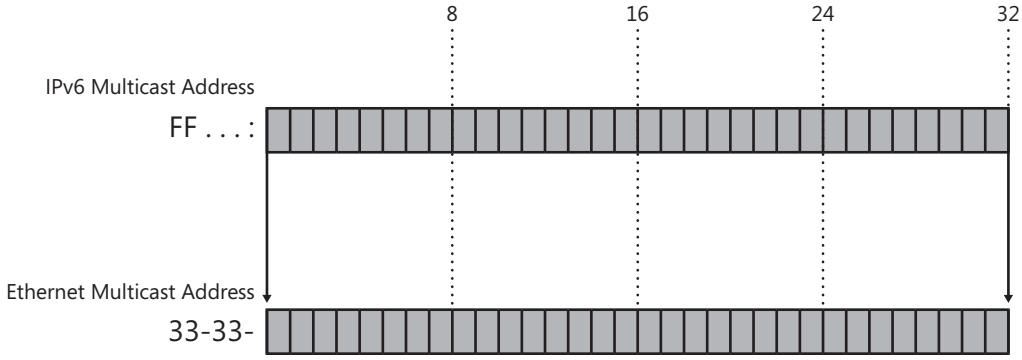

Figure 3-8 The mapping of IPv6 multicast addresses to Ethernet multicast addresses

Ethernet network adapters maintain a table of interesting destination MAC addresses. If an Ethernet frame with an interesting destination MAC address is received, it is passed to upper layers for additional processing. By default, this table contains the MAC-level broadcast address (0xFF-FF-FF-FF-FF-FF) and the unicast MAC address assigned to the adapter. To facilitate efficient delivery of multicast traffic, additional multicast destination addresses can be added or removed from the table. For every multicast address being listened to by the host, there is a corresponding entry in the table of interesting MAC addresses.

For example, an IPv6 host with the Ethernet MAC address of 00-AA-00-3F-2A-1C (link-local address of FE80::2AA:FF:FE3F:2A1C) adds the following multicast MAC addresses to the table of interesting destination MAC addresses on the Ethernet adapter:

- The address of 33-33-00-00-00-01, which corresponds to the link-local scope all-nodes multicast address of FF02::1 (fully expressed as FF02:0000:0000:0000:0000:0000:0000:0001).
- The address of 33-33-FF-3F-2A-1C, which corresponds to the solicited-node address of FF02::1:FF3F:2A1C. Remember that the solicited-node address is the prefix FF02::1:FF00:0/104 and the last 24 bits of the unicast IPv6 address.

Additional multicast addresses on which the host is listening are added and removed from the table as needed.

Anycast IPv6 Addresses

An anycast address is assigned to multiple interfaces. Packets addressed to an anycast address are forwarded by the routing infrastructure to the nearest interface to which the anycast address is assigned. To facilitate delivery, the routing infrastructure must be aware of the interfaces that have anycast addresses assigned to them and their distance in terms of routing metrics. This awareness is accomplished by the propagation of host routes throughout the routing infrastructure of the portion of the network that cannot summarize the anycast address using a route prefix.

For example, for the anycast address 3FFE:2900:D005:6187:2AA:FF:FE89:6B9A, host routes for this address are propagated within the routing infrastructure of the organization assigned the 48-bit prefix 3FFE:2900:D005::/48. Because a node assigned this anycast address can be placed anywhere on the organization's intranet, host routes for all nodes assigned this anycast address are needed in the routing tables of all routers within the organization. Outside the organization, this anycast address is summarized by the 3FFE:2900:D005::/48 prefix that is assigned to the organization. Therefore, the host routes needed to deliver IPv6 packets to the nearest anycast group member within an organization's intranet are not needed in the routing infrastructure of the IPv6 Internet.

As of RFC 4291, anycast addresses are used only as destination addresses and are assigned only to routers. Anycast addresses are assigned out of the unicast address space, and the scope of an anycast address is the scope of the type of unicast address from which the anycast address is assigned. It is not possible to determine if a given destination unicast address is also an anycast address. The only nodes that have this awareness are the routers that use host routes to forward the anycast traffic to the nearest anycast group member and the anycast group members themselves.

Subnet-Router Anycast Address

The Subnet-Router anycast address is defined in RFC 4291 and is required. It is created from the subnet prefix for a given interface. When the Subnet-Router anycast address is constructed, the bits in the subnet prefix are fixed at their appropriate values and the remaining bits are set to 0. Figure 3-9 shows the structure of the Subnet-Router anycast address.

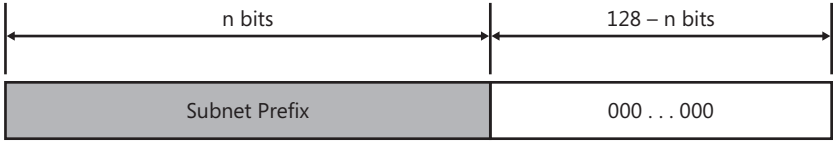

Figure 3-9 The structure of the Subnet-Router anycast address

All router interfaces attached to a subnet are assigned the Subnet-Router anycast address for that subnet. The Subnet-Router anycast address is used to communicate with the nearest router connected to a specified subnet.

IPv6 Addresses for a Host

An IPv4 host with a single network adapter typically has a single IPv4 address assigned to that adapter. An IPv6 host, however, usually has multiple IPv6 addresses assigned to each adapter. The interfaces on a typical IPv6 host are assigned the following unicast addresses:

- A link-local address for each interface
- Additional unicast addresses for each interface (which could be one or multiple unique local or global addresses)
- The loopback address (::1) for the loopback interface

Typical IPv6 hosts are always logically multihomed because they always have at least two addresses with which they can receive packets—a link-local address for local link traffic and a routable unique local or global address.

Additionally, each interface on an IPv6 host is listening for traffic on the following multicast addresses:

- The interface-local scope all-nodes multicast address (FF01::1)
- The link-local scope all-nodes multicast address (FF02::1)
- The solicited-node address for each unicast address assigned
- The multicast addresses of joined groups

IPv6 Addresses for a Router

The interfaces on an IPv6 router are assigned the following unicast addresses:

- A link-local address for each interface
- Additional unicast addresses for each interface (which could be one or multiple unique local or global addresses)
- The loopback address (::1) for the loopback interface

Additionally, the interfaces of an IPv6 router are assigned the following anycast addresses:

- A Subnet-Router anycast address for each subnet
- Additional anycast addresses (optional)

Additionally, the interfaces of an IPv6 router are listening for traffic on the following multicast addresses:

- The interface-local scope all-nodes multicast address (FF01::1)
- The interface-local scope all-routers multicast address (FF01::2)
- The link-local scope all-nodes multicast address (FF02::1)
- The link-local scope all-routers multicast address (FF02::2)
- The site-local scope all-routers multicast address (FF05::2)
- The solicited-node address for each unicast address assigned
- The multicast addresses of joined groups

Subnetting the IPv6 Address Space

Just as in IPv4, the IPv6 address space can be divided by using high-order bits that do not already have fixed values to create subnetted address prefixes. These are used either to summarize a level in the routing or addressing hierarchy (with a prefix length less than 64), or to define a specific subnet or network segment (with a prefix length of 64). IPv4 subnetting differs from IPv6 subnetting in the definition of the host ID portion of the address. In IPv4, the host ID can be of varying length, depending on the subnetting scheme. For currently defined unicast IPv6 addresses, the host ID is the interface ID portion of the IPv6 unicast address and is always a fixed size of 64 bits.

For most network administrators within an organization, subnetting the IPv6 address space consists of using subnetting techniques to divide the subnet ID portion of a global or unique local address prefix in a manner that allows for route summarization and delegation of the remaining address space to different portions of an IPv6 intranet. For both global and unique local addresses, the first 48 bits of the address are fixed. For the global address, the first 48 bits are fixed and allocated by an ISP. For the unique local address, the first 48 bits are fixed at FD00::/8 and the random 40-bit global ID assigned to a site of an organization.

Subnetting the subnet ID portion of a global or unique local address space requires a two-step procedure:

1. Determine the number of bits to be used for the subnetting.
2. Enumerate the new subnetted address prefixes.

The subnetting technique described here assumes that subnetting is done by dividing the 16-bit address space of the subnet ID using the high-order bits in the subnet ID. Although this method promotes hierarchical addressing and routing, it is not required. For example, in a small organization with a small number of subnets, you can also create a flat addressing space for the subnet ID by numbering the subnets starting at 0.

Step 1: Determining the Number of Subnetting Bits

The number of bits being used for subnetting determines the possible number of new subnetted address prefixes that can be allocated to portions of your network based on geographical or departmental divisions. In a hierarchical routing infrastructure, you need to determine how many address prefixes, and therefore how many bits, you need at each level in the hierarchy. The more bits you choose for the various levels of the hierarchy, the fewer bits you will have available to enumerate individual subnets in the last level of the hierarchy.

Depending on the needs of your organization, your subnetting scheme might be along nibble (hexadecimal digit) or bit boundaries. If you can subnet along nibble boundaries, your subnetting scheme becomes simplified and each hexadecimal digit can represent a level in the subnetting hierarchy. For example, a network administrator decides to implement a three-level hierarchy that uses the first nibble for an organization's campus, the next nibble for a building within a campus, and the last two nibbles for a subnet within a building. An example subnet ID for this scheme is 142A, which indicates campus 1, building 4, and subnet 42 (0x2A).

In some cases, bit-level subnetting is required. For example, a network administrator decides to implement a two-level hierarchy reflecting a geographical/departmental structure and uses 4 bits for the geographical level and 6 bits for the departmental level. This means that each department in each geographical location has only 6 bits of subnetting space left (16–4–6), or only 64 (= 2^6) subnets per department.

On any given level in the hierarchy, you will have a number of bits that are already fixed by the next level up in the hierarchy (*f*), a number of bits used for subnetting at the current level in the hierarchy (*s*), and a number of bits remaining for the next level down in the hierarchy (*r*). At all times, $f + s + r = 16$. This relationship is shown in Figure 3-10.

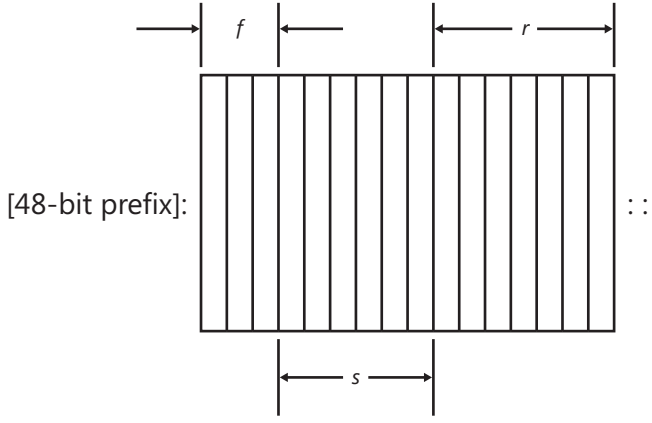

Figure 3-10 The subnetting of a subnet ID

Step 2: Enumerating Subnetted Address Prefixes

Based on the number of bits used for subnetting, you must determine the new subnetted address prefixes. There are three main approaches:

- **Binary** Enumerate new subnetted address prefixes by using binary representations of the subnet ID and converting to hexadecimal for the subnetted address prefixes.
- **Hexadecimal** Enumerate new subnetted address prefixes by using hexadecimal representations of the subnet ID and a calculated increment between successive subnetted address prefixes.
- **Decimal** Enumerate new subnetted address prefixes by using decimal representations of the subnet ID and increment.

Any of these methods produces the same result: an enumerated list of subnetted address prefixes.

Using the Binary Method

In the binary method, the 16-bit subnet ID is expressed as a 16-digit binary number. The bits within the subnet ID that are being used for subnetting are incremented for all their possible values, and for each value, the 16-digit binary number is converted to hexadecimal and combined with the 48-bit site prefix, producing the subnetted address prefixes.

1. Based on *s* (the number of bits chosen for subnetting), *m* (the prefix length of the address prefix being subnetted), and *f* (the number of bits already subnetted), calculate the following:

   $f = m - 48$

   f is the number of bits within the subnet ID that are already fixed.

   $n = 2^s$

   n is the number of address prefixes that are obtained.

   $l = 48 + f + s$

   l is the prefix length of the new subnetted address prefixes.
2. Create a three-column table with *n* entries. The first column is the address prefix number (starting with 1), the second column is the binary representation of the subnet ID portion of the new address prefix, and the third column is the subnetted address prefix (in hexadecimal), which includes the 48-bit site prefix and the subnet ID.
3. In the first table entry, set all the bits being used for subnetting to 0. Convert the resulting 16-digit binary number to hexadecimal, combine it with the 48-bit site prefix, and write the subnetted address prefix. This first subnetted address prefix is just the original address prefix with the new prefix length.
4. In the next table entry, increment the value within the subnet bits. Convert the 16-digit binary number to hexadecimal, combine it with the 48-bit site prefix, and write the resulting subnetted address prefix.
5. Repeat step 4 until the table is complete.

For example, to perform a 3-bit subnetting of the global address prefix 2001:DB8:0:C000::/51, we first calculate the values for the number of prefixes and the new prefix length. Our starting values are $s = 3$, and $f = 51 - 48 = 3$. The number of prefixes is 8 ($n = 2^3$). The new prefix length is 54 ($l = 48 + 3 + 3$). The initial value for the subnet ID in binary is 1100 0000 0000 0000 (0xC000 converted to binary).

Next, we construct a table with eight entries. The entry for the address prefix 1 is 2001:DB8:0:C000::/54. Additional entries are increments of the subnet bits in the subnet ID portion of the address prefix, as shown in Table 3-3.

Table 3-3 The Binary Subnetting Technique for Address Prefix 2001:DB8:0:C000::/51

Address Prefix Number	Binary Representation of Subnet ID	Subnetted Address Prefix
1	1100 0000 0000 0000	2001:DB8:0:C000::/54
2	1100 0100 0000 0000	2001:DB8:0:C400::/54
3	1100 1000 0000 0000	2001:DB8:0:C800::/54
4	1100 1100 0000 0000	2001:DB8:0:CC00::/54
5	1101 0000 0000 0000	2001:DB8:0:D000::/54
6	1101 0100 0000 0000	2001:DB8:0:D400::/54
7	1101 1000 0000 0000	2001:DB8:0:D800::/54
8	1101 1100 0000 0000	2001:DB8:0:DC00::/54

In Table 3-3, the underline in the second column shows the bits that are being used for subnetting.

Using the Hexadecimal Method

Although the binary method allows you to see how the subnetted address prefixes are determined at their most basic level, this method is laborious and does not scale well. For example, imagine performing an 8-bit subnetting using the binary method, producing 256 subnetted prefixes. For an arbitrary subnetting scheme, a more formulaic approach is needed. The following method uses a formula for computing the hexadecimal increment between successive subnetted address prefixes:

1. Based on *s* (the number of bits chosen for subnetting), *m* (the prefix length of the address prefix being subnetted), and *F* (the hexadecimal value of the subnet being subnetted), calculate the following:

   $f = m - 48$

   f is the number of bits within the subnet ID that are already fixed.

   $n = 2^s$

   n is the number of address prefixes that are obtained.

   $i = 2^{16-(f+s)}$

i is the incremental value between each successive subnet ID expressed in hexadecimal form.

$l = 48 + f + s$

l is the prefix length of the new subnetted address prefixes.

2. Create a two-column table with *n* entries. The first column is the address prefix number (starting with 1), and the second column is the new subnetted address prefix.
3. In the first table entry, based on *F*, the hexadecimal value of the subnet ID being subnetted, set the subnetted address prefix to *48-bit prefix*:*F*::/*l*.
4. In the next table entry, increase the value within the subnet ID portion of the site address by *i*. For example, in the second table entry, set the subnetted prefix to *48-bit prefix*:*F* + *i*::/*l*.
5. Repeat step 4 until the table is complete.

For example, to perform a 3-bit subnetting of the global address prefix 2001:DB8:0:C000::/51, we first calculate the values of the number of prefixes, the increment, and the new prefix length. Our starting values are *F* = 0xC000, *s* = 3, and *f* = 51 − 48 = 3. The number of prefixes is 8 ($n = 2^3$). The increment is 0x400 ($i = 2^{16-(f+s)}$ = 1024 = 0x400). The new prefix length is 54 (*l* = 48 + 3 + 3).

Next, we construct a table with eight entries. The entry for the address prefix 1 is 2001:DB8:0:C000::/54. Additional entries in the table are successive increments of *i* in the subnet ID portion of the address prefix, as shown in Table 3-4.

Table 3-4 The Hexadecimal Subnetting Technique for Address Prefix 2001:DB8:0:C000::/51

Address Prefix Number	Subnetted Address Prefix
1	2001:DB8:0:C000::/54
2	2001:DB8:0:C400::/54
3	2001:DB8:0:C800::/54
4	2001:DB8:0:CC00::/54
5	2001:DB8:0:D000::/54
6	2001:DB8:0:D400::/54
7	2001:DB8:0:D800::/54
8	2001:DB8:0:DC00::/54

Using the Decimal Method

If you are more comfortable working with decimal numbers, the following formulaic procedure will produce the same results. However, there are additional steps to convert to decimal and then back to hexadecimal for the representation of the subnetted address prefix.

1. Based on *s* (the number of bits chosen for subnetting), *m* (the prefix length of the address prefix being subnetted), and *F* (the hexadecimal value of the subnet ID being subnetted), calculate the following:

   $f = m - 48$

   f is the number of bits within the subnet ID that are already fixed.

   $n = 2^s$

   n is the number of address prefixes that are obtained.

   $i = 2^{16-(f+s)}$

   i is the incremental value between each successive subnet ID.

   $l = 48 + f + s$

   l is the prefix length of the new subnetted address prefixes.

   D = decimal representation of F

2. Create a three-column table with *n* entries. The first column is the address prefix number (starting with 1), the second column is the decimal representation of the subnet ID portion of the new address prefix, and the third column is the new subnetted address prefix.
3. In the first table entry, the decimal representation of the subnet ID is *D* and the subnetted address prefix is *48-bit prefix*:*F*::/*l*.
4. In the next table entry, for the second column, increase the value of the decimal representation of the subnet ID by *i*. For example, in the second table entry, the decimal representation of the subnet ID is *D* + *i*.
5. For the third column, convert the decimal representation of the subnet ID to hexadecimal and construct the prefix from *48-bit prefix*:*subnet ID*::/*l*. For example, in the second table entry, the subnetted address prefix is *48-bit prefix*:*D* + *i* (converted to hexadecimal)::/*l*.
6. Repeat steps 4 and 5 until the table is complete.

For example, to perform a 3-bit subnetting of the global address prefix 2001:DB8:0:C000::/51, we first calculate the values of the number of prefixes, the increment, the new prefix length, and the decimal representation of the starting subnet ID. Our starting values are *F* = 0xC000, *s* = 3, and *f* = 51 − 48 = 3. The number of prefixes is 8 ($n = 2^3$). The increment is 1024 ($i = 2^{16-(f+s)}$). The new prefix length is 54 (*l* = 48 + 3 + 3). The decimal representation of the starting subnet ID is 49152 (*D* = 0xC000 = 49152).

Next, we construct a table with eight entries. The entry for the address prefix 1 is 49152 and 2001:DB8:0:C000::/54. Additional entries in the table are successive increments of *i* in the subnet ID portion of the address prefix, as shown in Table 3-5.

Table 3-5 The Decimal Subnetting Technique for Address Prefix 2001:DB8:0:C000::/51

Address Prefix Number	Decimal Representation of Subnet ID	Subnetted Address Prefix
1	49152	2001:DB8:0:C000::/54
2	50176	2001:DB8:0:C400::/54
3	51200	2001:DB8:0:C800::/54
4	52224	2001:DB8:0:CC00::/54
5	53248	2001:DB8:0:D000::/54
6	54272	2001:DB8:0:D400::/54
7	55296	2001:DB8:0:D800::/54
8	56320	2001:DB8:0:DC00::/54

IPv6 Interface Identifiers

The last 64 bits of a currently defined IPv6 unicast address are for the interface identifier, which is unique for a 64-bit subnet prefix of a unicast IPv6 address. In IPv4, the host or node ID portion of an IPv4 address is a logical identifier of an interface on an IPv4 subnet. IPv4 host IDs are of variable length, depending on the subnetting scheme and how many interfaces you want to allow on a given subnet. For example, with an 8-bit host ID, there were $2^8 - 2$ or 254 possible host IDs. (The all-zeros and all-ones combinations are reserved.)

In IPv6, the interface ID is of fixed length. This length was not fixed at 64 bits to allow up to 2^{64} possible hosts on the same subnet. Rather, the IPv6 interface ID is 64 bits long to accommodate the mapping of current 48-bit MAC addresses used by many local area network (LAN) technologies such as Ethernet and the mapping of 64-bit MAC addresses of IEEE 1394 (also known as FireWire) and future LAN technologies.

The ways in which an interface identifier for a LAN interface is determined are the following:

- As defined in RFC 4291, it can be derived from the Extended Unique Identifier (EUI)-64 address. The 64-bit EUI-64 address is defined by the Institute of Electrical and Electronics Engineers (IEEE). EUI-64 addresses are either assigned to a network adapter or derived from IEEE 802 addresses. This is the default behavior for IPv6 in Windows XP and Windows Server 2003.
- As defined in RFC 4941, it might have a temporarily assigned, randomly generated interface identifier to provide a level of anonymity. For more information, see the "Temporary Address Interface Identifiers" section in this chapter.
- It is assigned during stateful address autoconfiguration—for example, via Dynamic Host Configuration Protocol for IPv6 (DHCPv6).
- As defined in RFC 5072, an interface identifier can be based on link-layer addresses or serial numbers, or it can be randomly generated when configuring a Point-to-Point Protocol (PPP) interface and an EUI-64 address is not available.

- It is assigned during manual address configuration.
- It is a permanent interface identifier that is randomly generated to mitigate address scans of unicast IPv6 addresses on a subnet. This is the default behavior for LAN interfaces for IPv6 in Windows Vista and Windows Server 2008. You can disable this behavior with the **netsh interface ipv6 set global randomizeidentifiers=disabled** command. When this behavior is disabled, IPv6 for Windows Vista and Windows Server 2008 will use EUI-64–based interface identifiers.

EUI-64 Address-Based Interface Identifiers

One way to derive an IPv6 interface identifier is through the EUI-64 address, a new type of MAC address for network adapters. To gain an understanding of EUI-64 addresses, it is useful to review the current MAC address format known as *IEEE 802 addresses*.

IEEE 802 Addresses

Network adapters for common LAN technologies such as Ethernet and IEEE 802.11 use a 48-bit address called an IEEE 802 address. It consists of a 24-bit company ID (also called the *manufacturer ID*) and a 24-bit extension ID (also called the *board ID*). The combination of the company ID, which is uniquely assigned to each manufacturer of network adapters, and the extension ID, which is uniquely assigned to each network adapter at the time of manufacture, produces a globally unique 48-bit address. This 48-bit address is also called the physical, hardware, or MAC address.

Figure 3-11 shows the structure of the 48-bit IEEE 802 address for Ethernet.

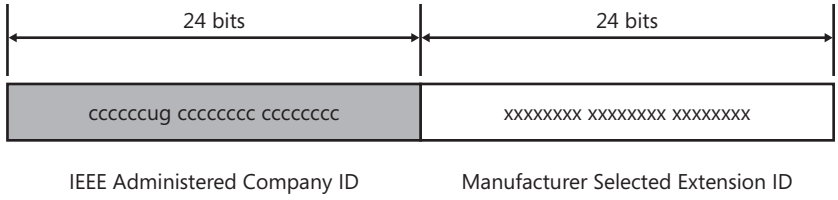

Figure 3-11 The structure of the 48-bit IEEE 802 address for Ethernet

Defined bits within the IEEE 802 address for Ethernet are as follows:

- **Universal/Local (U/L)** The seventh bit in the first byte is used to indicate whether the address is universally or locally administered. If the U/L bit is set to 0, the IEEE (through the designation of a unique company ID) has administered the address. If the U/L bit is set to 1, the address is locally administered. In this case, the network administrator has overridden the manufactured address and specified a different address. The U/L bit is designated by the **u** in Figure 3-11.
- **Individual/Group (I/G)** The eighth (low-order) bit of the first byte is used to indicate whether the address is an individual address (unicast) or a group address (multicast).

When set to 0, the address is a unicast address. When set to 1, the address is a multicast address. The I/G bit is designated by the **g** in Figure 3-11.

For a typical IEEE 802 address assigned to a network adapter, both the U/L and I/G bits are set to 0, corresponding to a universally administered, unicast MAC address.

IEEE EUI-64 Addresses

The IEEE EUI-64 address represents a new standard for network interface addressing. The company ID is still 24-bits long, but the extension ID is 40 bits, creating a much larger address space for a network adapter manufacturer. The EUI-64 address uses the U/L and I/G bits in the same way as the IEEE 802 address.

Figure 3-12 shows the structure of the EUI-64 address.

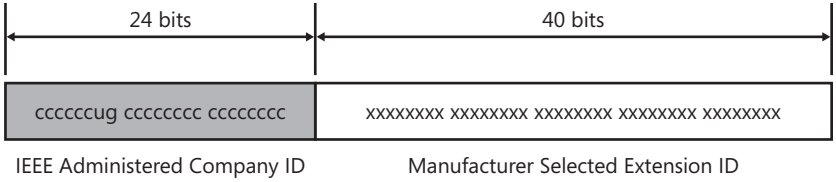

Figure 3-12 The structure of the EUI-64 address

Mapping IEEE 802 Addresses to EUI-64 Addresses To create an EUI-64 address from an IEEE 802 address, the 16 bits of 11111111 11111110 (0xFFFE) are inserted into the IEEE 802 address between the company ID and the extension ID, as shown in Figure 3-13.

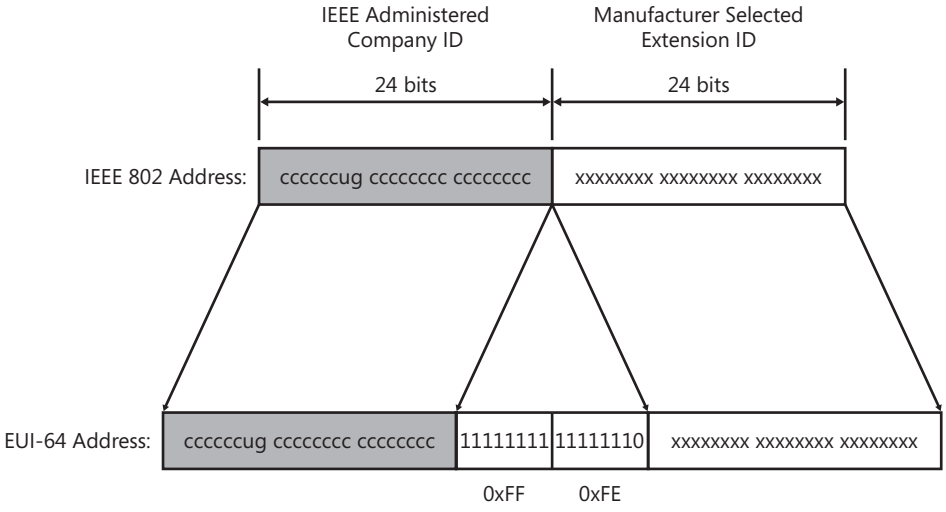

Figure 3-13 The mapping of IEEE 802 addresses to EUI-64 addresses

Obtaining Interface Identifiers for IPv6 Addresses

To obtain the 64-bit interface identifier for IPv6 unicast addresses, the U/L bit in the EUI-64 address is complemented. (If it is a 1 in the EUI-64 address, it is set to 0; and if it is a 0 in the EUI-64 address, it is set to 1.)

The main reason for complementing the U/L bit is to provide greater compressibility of locally administered EUI-64 addresses. It is common practice when assigning locally administered addresses to number them in a simple way. For example, on a point-to-point link, you can assign to one interface on the link the locally administered EUI-64 address of 02-00-00-00-00-00-00-01 and to the other interface the locally administered EUI-64 address of 02-00-00-00-00-00-00-02. If the U/L bit is not complemented, the corresponding link-local addresses for these two interfaces become FE80::200:0:0:1 and FE80::200:0:0:2. By complementing the U/L bit, the corresponding link-local addresses for these two interfaces become FE80::1 and FE80::2.

Figure 3-14 shows the conversion of an EUI-64 address to an IPv6 interface identifier.

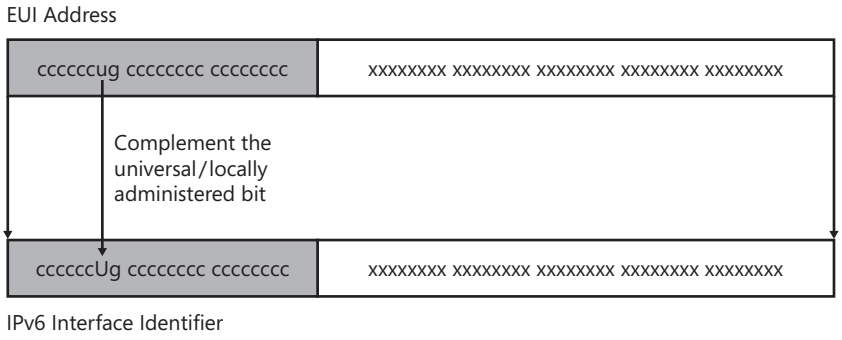

Figure 3-14 The conversion of an EUI-64 address to an IPv6 interface identifier

> **Note** Because the U/L bit is complemented when converting an EUI-64 address to an IPv6 interface identifier, the resulting bit in the IPv6 interface identifier has the opposite interpretation of the IEEE-defined U/L bit. If the seventh bit of the IPv6 interface identifier is set to 0, it is locally administered. If the seventh bit of the IPv6 interface identifier is set to 1, it is universally administered.

Converting IEEE 802 Addresses to IPv6 Interface Identifiers To obtain an IPv6 interface identifier from an IEEE 802 address, you must first map the IEEE 802 address to an EUI-64 address, and then complement the U/L bit. Figure 3-15 shows this conversion process for a universally administered, unicast IEEE 802 address.

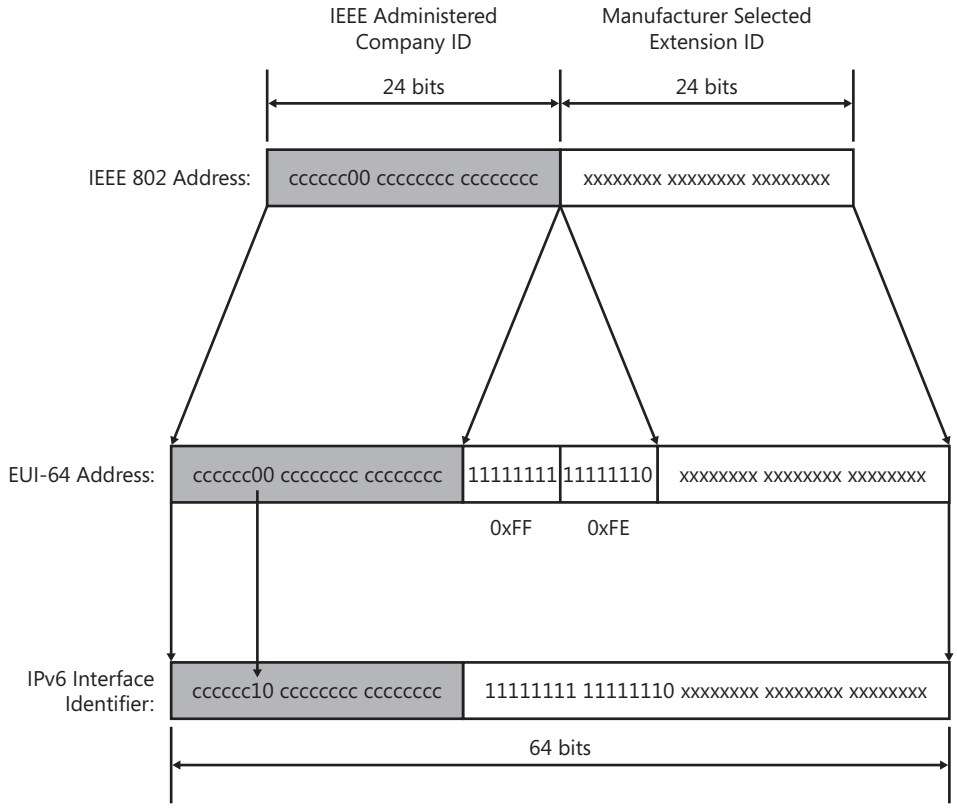

Figure 3-15 The conversion of an IEEE 802 address to an IPv6 interface identifier

IEEE 802 Address Conversion Example Host A has the Ethernet MAC address of 00-AA-00-3F-2A-1C. First, it is converted to EUI-64 format by inserting FF-FE between the third and fourth bytes, yielding 00-AA-00-FF-FE-3F-2A-1C. Then, the U/L bit, which is the seventh bit in the first byte, is complemented. The first byte in binary form is 00000000. When the seventh bit is complemented, it becomes 00000010 (0x02). The final result is 02-AA-00-FF-FE-3F-2A-1C which, when converted to colon hexadecimal notation, becomes the interface identifier 2AA:FF:FE3F:2A1C. As a result, the link-local address that corresponds to the network adapter with the MAC address of 00-AA-00-3F-2A-1C is FE80::2AA:FF:FE3F:2A1C.

Note When complementing the U/L bit, add 0x2 to the first byte if the EUI-64 address is universally administered, and subtract 0x2 from the first byte if the EUI-64 address is locally administered.

Temporary Address Interface Identifiers

In today's IPv4-based Internet, a typical Internet user dials an ISP and obtains an IPv4 address using PPP and the Internet Protocol Control Protocol (IPCP). Each time the user dials, a different IPv4 address might be obtained. Therefore, it is not easy to track a dial-up user's traffic on the Internet based on the user's IPv4 address.

For IPv6-based dial-up connections, the user is assigned a 64-bit prefix, at the time of connection, by using router discovery, which consists of an exchange of Router Solicitation and Router Advertisement messages. If the interface identifier is always based on the EUI-64 address (as derived from the static IEEE 802 address), it is possible to identify the traffic of a specific node regardless of the prefix assigned at the time of connection. The use of the same 64-bit interface identifier allows identification of a user's traffic whether the user is accessing the Internet from home or from work. This makes it easy for Internet merchants and malicious users to track a specific user and his or her use of the Internet.

To address this concern and provide the same level of anonymity as that provided with IPv4, RFC 4941 describes an alternative derivation of the IPv6 interface identifier that is randomly generated and changes over time.

The initial interface identifier is generated using random number techniques. For IPv6 systems that do not have the ability to store any history information for generating future values of the interface identifier, a new random interface identifier is generated each time the IPv6 protocol is initialized. For IPv6 systems that do have storage capabilities, a history value is stored and when the IPv6 protocol is initialized, a new interface identifier is created through the following process:

1. Retrieve the history value from storage, and append the interface identifier based on the EUI-64 address of the adapter.
2. Compute the Message Digest-5 (MD5) hash over the quantity in step 1. The MD5 hash computation will produce a 128-bit value.
3. Store the low-order 64 bits of the MD5 hash computed in step 2 as the history value for the next computation of the interface identifier.
4. Take the high-order 64 bits of the MD5 hash computed in step 2 and set the seventh bit to zero. The seventh bit corresponds to the U/L bit, which, when set to 0, indicates a locally administered interface identifier. The result is the interface identifier.

The resulting IPv6 address, based on this random interface identifier, is known as a *temporary address*. Temporary addresses are generated for public address prefixes that use stateless address autoconfiguration. Temporary addresses are used for the lower of the following values of the valid and preferred lifetimes:

- The lifetimes included in the Prefix Information option in the received Router Advertisement message.
- Local default values of 1 week for valid lifetime and 1 day for preferred lifetime.

After the temporary address valid lifetime expires, a new interface identifier and temporary address is generated. For more information about router discovery, see Chapter 6, "Neighbor Discovery." For more information about stateless address autoconfiguration and valid and preferred lifetimes, see Chapter 8.

IPv4 Addresses and IPv6 Equivalents

To summarize the relationships between IPv4 addressing and IPv6 addressing, Table 3-6 lists both IPv4 addresses and addressing concepts and their IPv6 equivalents.

Table 3-6 IPv4 Addressing Concepts and Their IPv6 Equivalents

IPv4 Address	IPv6 Address
Internet address classes	Not applicable in IPv6
Multicast addresses (224.0.0.0/4)	IPv6 multicast addresses (FF00::/8)
Broadcast addresses	Not applicable in IPv6
Unspecified address is 0.0.0.0	Unspecified address is ::
Loopback address is 127.0.0.1	Loopback address is ::1
Public IP addresses	Global unicast addresses
Private IP addresses (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16)	Unique local (FD00::/8) or site-local addresses (FEC0::/10) (deprecated)
APIPA addresses (169.254.0.0/16)	Link-local addresses (FE80::/64)
Text representation: Dotted-decimal notation	Text representation: Colon hexadecimal format with suppression of leading zeros and zero compression.
Prefix representation: Subnet mask in dotted-decimal notation or prefix length notation	Prefix representation: Prefix length notation only

References

The following references were cited in this chapter:

- RFC 3306 — "Unicast-Prefix-based IPv6 Multicast Addresses"
- RFC 3587 — "IPv6 Global Unicast Address Format"
- RFC 3879 — "Deprecating Site Local Addresses"
- RFC 3927 — "Dynamic Configuration of IPv4 Link-Local Addresses"
- RFC 3956 — "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address"
- RFC 4007 — "IPv6 Scoped Address Architecture"
- RFC 4193 — "Unique Local IPv6 Unicast Addresses"

- RFC 4291 — "IP Version 6 Addressing Architecture"
- RFC 4941 — "Privacy Extensions for Stateless Address Autoconfiguration in IPv6"
- RFC 5072 — "IP Version 6 over PPP"

You can obtain these RFCs from the \RFCs_and_Drafts folder on the companion CD-ROM or from *http://www.ietf.org/rfc.html*.

Testing for Understanding

To test your understanding of IPv6 addressing, answer the following questions. See Appendix D, "Testing for Understanding Answers," to check your answers.

1. Why is the IPv6 address length 128 bits?
2. Express FEC0:0000:0000:0001:02AA:0000:0000:0007A more efficiently.
3. How many blocks and bits are expressed by "::" in the addresses 2001:DB8::2AA:9FF:FE56:24DC and FF02::2?
4. Describe the difference between unicast, multicast, and anycast addresses in terms of a host sending packets to zero or more interfaces.
5. Why are no broadcast addresses defined for IPv6?
6. Define the structure, including field sizes, of the global unicast address.
7. Define the scope for each of the different types of unicast addresses.
8. Explain how global and unique local addressing can share the same subnetting infrastructure within an organization.
9. Define the structure, including field sizes, of the multicast address.
10. Explain how the solicited-node multicast address acts as a pseudo-unicast address.
11. How do routers know the nearest location of an anycast group member?
12. Perform a 4-bit subnetting on the unique local prefix FD1A:39C1:4BC2:3D80::/57.
13. What is the EUI-64–based IPv6 interface identifier for the universally administered, unicast IEEE 802 address of 0C-1C-09-A8-F9-CE? What is the corresponding link-local address? What is the corresponding solicited-node multicast address?
14. What is the IPv6 interface identifier for the locally administered, unicast EUI-64 address of 02-00-00-00-00-00-00-09? What is the corresponding link-local address?
15. For each type of address shown in the following table, identify how the address begins in colon hexadecimal notation.

Type of Address	Begins with…
Link-local unicast address	FE80
Site-local unicast address	
Unique local unicast address	
Global address (as defined by RFC 3587)	
Multicast address	
Link-local scope multicast address	
Site-local scope multicast address	
Solicited-node multicast address	
IPv4-mapped address	
6to4 address	
Teredo address	

Chapter 4

The IPv6 Header

At the end of this chapter, you should be able to do the following:

- Describe the structure of an IPv6 packet.
- List and describe the fields in the IPv4 header.
- List and describe the fields in the IPv6 header.
- Compare and contrast the fields in the IPv4 header with the fields in the IPv6 header.
- List and describe each IPv6 extension header.
- Describe the IPv6 maximum transmission unit (MTU).
- Describe the new pseudo-header used for upper-layer checksums.

Structure of an IPv6 Packet

An Internet Protocol version 6 (IPv6) packet consists of an IPv6 header, extension headers, and an upper-layer protocol data unit. Figure 4-1 shows the structure of an IPv6 packet.

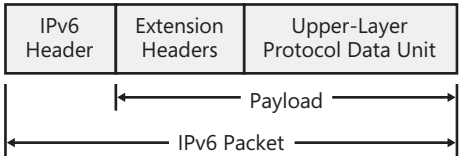

Figure 4-1 The structure of an IPv6 packet

The components of an IPv6 packet are the following:

- **IPv6 Header** The IPv6 header is always present and is a fixed size of 40 bytes. The fields in the IPv6 header are described in the "IPv6 Header" section in this chapter.
- **Extension Headers** Zero or more extension headers can be present and are of varying lengths. If extension headers are present, a Next Header field in the IPv6 header indicates the first extension header. Within each extension header is another Next Header field, indicating the next extension header. The last extension header indicates the header for the upper-layer protocol—such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or Internet Control Message Protocol for version 6 (ICMPv6)—contained within the upper-layer protocol data unit.

  The IPv6 header and extension headers replace the existing IPv4 header and its options. The new extension header format allows IPv6 to be enhanced to support future needs and capabilities. Unlike options in the IPv4 header, IPv6 extension headers have no

maximum size and can expand to accommodate all the extension data needed for IPv6 communication. IPv6 extension headers are described in the "IPv6 Extension Headers" section in this chapter.

- **Upper-Layer Protocol Data Unit** The upper-layer protocol data unit (PDU) typically consists of an upper-layer protocol header and its payload (for example, an ICMPv6 message, a TCP segment, or a UDP message).

  The IPv6 packet payload is the combination of the IPv6 extension headers and the upper-layer PDU. Normally, it can be up to 65,535 bytes long. IPv6 packets with payloads larger than 65,535 bytes in length, known as *jumbograms*, can also be sent.

IPv4 Header

Before examining the IPv6 header, you might find it helpful, for contrasting purposes, to review the IPv4 header shown in Figure 4-2.

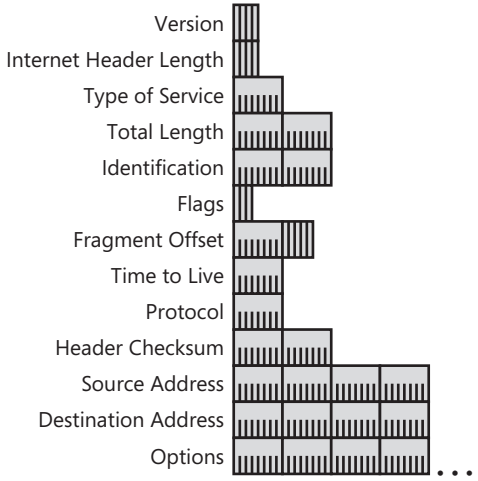

Figure 4-2 The structure of the IPv4 header

Following is a list of the fields in the IPv4 header:

- **Version** The Version field indicates the version of IP and is set to 4. The size of this field is 4 bits.
- **Internet Header Length** The Internet Header Length (IHL) field indicates the number of 4-byte blocks in the IPv4 header. The size of this field is 4 bits. Because an IPv4 header is a minimum of 20 bytes in size, the smallest value of the IHL field is 5. IPv4 options can extend the minimum IPv4 header size in increments of 4 bytes. If an IPv4 option is not an integral multiple of 4 bytes in length, the remaining bytes are padded with padding options, making the entire IPv4 header an integral multiple of 4 bytes. With a maximum IHL value of 0xF, the maximum size of the IPv4 header, including options, is 60 bytes (15 × 4).

- **Type of Service** The Type of Service field indicates the desired service expected by this packet for delivery through routers across the IPv4 internetwork. The size of this field is 8 bits, including bits originally defined in RFC 791 for precedence, delay, throughput, reliability, and cost characteristics. RFC 2474 provides the modern definition as the Differentiated Services (DS) field. The high-order 6 bits of the DS field comprise the DS Code Point (DSCP) field. The DSCP field allows devices in a network to mark, unmark, and classifypackets for forwarding. This is usually done based on the needs of an application. For example, Voice over IP and other real-time packets take precedence over e-mail in congested areas of the network. This is commonly referred to as Quality of Service (QoS). The low-order 2 bits of the Type of Service field are used for Explicit Congestion Notification (ECN), as defined in RFC 3168.
- **Total Length** The Total Length field indicates the total length of the IPv4 packet (IPv4 header + IPv4 payload) and does not include link-layer framing. The size of this field is 16 bits, which can indicate an IPv4 packet that is up to 65,535 bytes long.
- **Identification** The Identification field identifies this specific IPv4 packet. The size of this field is 16 bits. The Identification field is selected by the source node of the IPv4 packet. If the IPv4 packet is fragmented, all the fragments retain the Identification field value so that the destination node can group the fragments for reassembly.
- **Flags** The Flags field identifies flags for the fragmentation process. The size of this field is 3 bits; however, only 2 bits are defined for current use. There are two flags—one to indicate whether the IPv4 packet can be fragmented and another to indicate whether more fragments follow the current fragment.
- **Fragment Offset** The Fragment Offset field indicates the position of the fragment relative to the beginning of the original IPv4 payload. The size of this field is 13 bits.
- **Time-to-Live** The Time-to-Live (TTL) field indicates the maximum number of links on which an IPv4 packet can travel before being discarded. The size of this field is 8 bits. The TTL field was originally defined as a time count for the number of seconds the packet could exist on the network. An IPv4 router determined the length of time required (in seconds) to forward the IPv4 packet and decremented the TTL accordingly. Modern routers almost always forward an IPv4 packet in less than a second, and they are required by RFC 791 to decrement the TTL by at least one. Therefore, the TTL becomes a maximum link count with the value set by the sending node. When the TTL equals 0, an ICMPv4 Time Exceeded-Time to Live Exceeded in Transit message is sent to the source of the packet and the packet is discarded.
- **Protocol** The Protocol field identifies the upper-layer protocol. The size of this field is 8 bits. For example, a value of 6 in this field identifies TCP as the upper-layer protocol, a decimal value of 17 identifies UDP, and a value of 1 identifies ICMPv4. The Protocol field is used to identify the upper-layer protocol that is to receive the IPv4 packet payload.

- **Header Checksum** The Header Checksum field provides a checksum on the IPv4 header only. The size of this field is 16 bits. The IPv4 payload is not included in the checksum calculation, as the IPv4 payload usually contains its own checksum. Each IPv4 node that receives IPv4 packets verifies the IPv4 header checksum and silently discards the IPv4 packet if checksum verification fails. When a router forwards an IPv4 packet, it must decrement the TTL. Therefore, the Header Checksum value is recomputed at each hop between source and destination.
- **Source Address** The Source Address field stores the IPv4 address of the originating host. The size of this field is 32 bits.
- **Destination Address** The Destination Address field stores the IPv4 address of an intermediate destination (in the case of source routing) or the destination host. The size of this field is 32 bits.
- **Options** The Options field stores one or more IPv4 options. The size of this field is a multiple of 32 bits (4 bytes). If an IPv4 option does not use all 32 bits, padding options must be added so that the IPv4 header is an integral number of 4-byte blocks that can be indicated by the IHL field.

IPv6 Header

The IPv6 header is a streamlined version of the IPv4 header. It eliminates fields that are either unneeded or rarely used, and it adds a field that provides better support for real-time traffic. Figure 4-3 shows the structure of the IPv6 header as described in RFC 2460.

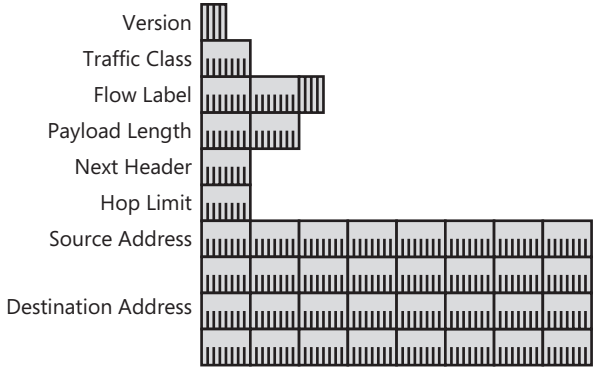

Figure 4-3 The structure of the IPv6 header

The following list describes the fields in the IPv6 header:

- **Version** The Version field indicates the version of IP and is set to 6. The size of this field is 4 bits. While the purpose of the Version field is defined in the same way for both IPv4 and IPv6, its value is not used to pass the packet to an IPv4 or IPv6 protocol layer. This identification is done through a protocol identification field in the link-layer header. For example, a common link-layer encapsulation for Ethernet, called Ethernet II, uses a

16-bit EtherType field to identify the Ethernet frame payload. For IPv4 packets, the EtherType field is set to 0x800. For IPv6 packets, the EtherType field is set to 0x86DD. Thus, the determination of the protocol of the Ethernet payload occurs before the packet is passed to the appropriate protocol layer.

- **Traffic Class** The Traffic Class field indicates the IPv6 packet's class or priority. The size of this field is 8 bits. This field provides functionality similar to the IPv4 Type of Service field. Like the Type of Service field in the IPv4 header, the first 6 bits of the Traffic Class field are the DSCP field as defined in RFC 2474 and the last 2 bits are used for ECN as defined in RFC 3168.
- **Flow Label** The Flow Label field indicates that this packet belongs to a specific sequence of packets between a source and destination, requiring special handling by intermediate IPv6 routers. The size of this field is 20 bits. The flow label is used for prioritized delivery, such as delivery needed by real-time data (voice and video). For default router handling, the Flow Label field is set to 0. To distinguish a given flow, an intermediate router can use the packet's source address, destination address, and flow label. Therefore, there can be multiple flows between a source and destination, as distinguished by separate non-zero flow labels. The details of the use of the Flow Label field are described in RFC 3697.
- **Payload Length** The Payload Length field indicates the length of the IPv6 payload. The size of this field is 16 bits. The Payload Length field includes the extension headers and the upper-layer PDU. With 16 bits, an IPv6 payload of up to 65,535 bytes can be indicated. For payload lengths greater than 65,535 bytes, the Payload Length field is set to 0 and the Jumbo Payload option is used in the Hop-by-Hop Options extension header, which is described in the "Hop-by-Hop Options Header" section in this chapter.
- **Next Header** The Next Header field indicates either the type of the first extension header (if present) or the protocol in the upper-layer PDU (such as TCP, UDP, or ICMPv6). The size of this field is 8 bits. When indicating an upper-layer protocol, the Next Header field uses the same values that are used in the IPv4 Protocol field.
- **Hop Limit** The Hop Limit field indicates the maximum number of links over which the IPv6 packet can travel before being discarded. The size of this field is 8 bits. The Hop Limit field is similar to the IPv4 TTL field, except that there is no historical relation to the amount of time (in seconds) that the packet is queued at the router. When Hop Limit equals 0 at a router, the router sends an ICMPv6 Time Exceeded-Hop Limit Exceeded in Transit message to the source and discards the packet.
- **Source Address** The Source Address field indicates the IPv6 address of the originating host. The size of this field is 128 bits.
- **Destination Address** The Destination Address field indicates the IPv6 address of the current destination node. The size of this field is 128 bits. In most cases, the Destination Address field is set to the final destination address. However, if a Routing extension header is present, the Destination Address field might be set to the address of the next intermediate destination.

Network Monitor Capture

Here is an example of an IPv6 header, as displayed by Network Monitor 3.1 (capture 04_01 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 40
  - Versions: IPv6, Internet Protocol, DSCP 0
     Version:  (0110..........................) IPv6, Internet Protocol, 6(0x6)
     DSCP:     (....000000......................) Differentiated services codepoint 0
     ECT:      (.........0......................) ECN-Capable Transport not set
     CE:       (..........0.....................) ECN-CE not set
     FlowLabel: (............00000000000000000000) 0
    PayloadLength: 40 (0x28)
    NextProtocol: ICMPv6, 58(0x3a)
    HopLimit: 128 (0x80)
    SourceAddress: FE80:0:0:0:260:97FF:FE02:6E8F
    DestinationAddress: FE80:0:0:0:260:97FF:FE02:6D3D
+ Icmpv6: Echo request, ID = 0x0, Seq = 0x18
```

This ICMPv6 Echo Request packet uses the default Traffic Class and Flow Label and a Hop Limit of 128, and it is sent between two hosts using link-local addresses.

Values of the Next Header Field

Table 4-1 lists typical values of the Next Header field for an IPv6 header or an IPv6 extension header. Each of the IPv6 extension headers is covered later in the chapter.

Table 4-1 Typical Values of the Next Header Field

Value (Decimal)	Header
0	Hop-by-Hop Options header
6	TCP
17	UDP
41	Encapsulated IPv6 header
43	Routing header
44	Fragment header
50	Encapsulating Security Payload header
51	Authentication header
58	ICMPv6
59	No next header
60	Destination Options header

For the most current list of the reserved values for the IPv4 Protocol and IPv6 Next Header fields, see *http://www.iana.org/assignments/protocol-numbers*.

In looking at the value of the Next Header field to indicate no next header, it would seem to make more sense to set its value to 0, rather than 59. However, the designers of IPv6 wanted to optimize the processing of IPv6 packets at intermediate routers. The only extension header that must be processed at every intermediate router is the Hop-by-Hop Options header. To optimize the test of whether the Hop-by-Hop Options header is present, its Next Header value is set to 0. In router hardware, it is easier to test for a value of 0 than to test for a value of 59.

Comparing the IPv4 and IPv6 Headers

In comparing the IPv4 and IPv6 headers, you can see the following:

- The number of fields has dropped from 12 (including options) in the IPv4 header to 8 in the IPv6 header.
- The number of fields that must be processed by an intermediate router has dropped from 6 to 4, making the forwarding of normal IPv6 packets more efficient.
- Seldom-used fields such as fields supporting fragmentation and options in the IPv4 header have been moved to extension headers in the IPv6 header.
- The size of the IPv6 header has doubled from 20 bytes for a minimum-sized IPv4 header to 40 bytes. However, the new IPv6 header contains source and destination addresses that are four times longer than IPv4 source and destination addresses.

Table 4-2 lists the individual differences between the IPv4 and IPv6 header fields.

Table 4-2 IPv4 Header Fields and Corresponding IPv6 Equivalents

IPv4 Header Field	IPv6 Header Field
Version	Same field but with a different version number.
Internet Header Length	Removed in IPv6. IPv6 does not include a Header Length field because the IPv6 header is always a fixed length of 40 bytes. Each extension header is either a fixed length or indicates its own length.
Type of Service	Replaced by the IPv6 Traffic Class field.
Total Length	Replaced by the IPv6 Payload Length field, which indicates only the size of the payload.
Identification Flags Fragment Offset	Removed in IPv6. Fragmentation information is not included in the IPv6 header. It is contained in a Fragment extension header.
Time-to-Live	Replaced by the IPv6 Hop Limit field.
Protocol	Replaced by the IPv6 Next Header field.
Header Checksum	Removed in IPv6. The link layer has a checksum that performs bit-level error detection for the entire IPv6 packet.
Source Address	The field is the same except that IPv6 addresses are 128 bits in length.
Destination Address	The field is the same except that IPv6 addresses are 128 bits in length.
Options	Removed in IPv6. IPv6 extension headers replace IPv4 options.

The one new field in the IPv6 header that is not included in the IPv4 header is the Flow Label field.

The result of the new IPv6 header is a reduction in the critical router loop, the set of instructions that must be executed to determine how to forward a packet. To forward a normal IPv4 packet, a router typically performs the following in its critical router loop:

1. Verify the Header Checksum field by performing its own checksum calculation and comparing its result with the result stored in the IPv4 header. Although this step is required by RFC 1812, modern high-speed routers commonly skip it.
2. Verify the value of the Version field. Although this step is not required by RFC 791 or 1812, performing this step saves network bandwidth, as a packet containing an invalid version number is not propagated across the IPv4 internetwork only to be discarded by the destination node.
3. Decrement the value of the TTL field. If its new value is less than 1, send an ICMPv4 Time Exceeded-Time to Live Exceeded in Transit message to the source of the packet and then discard the packet. If not, place the new value in the TTL field.
4. Check for the presence of IPv4 header options. If present, process them.
5. Use the value of the Destination Address field and the contents of the local routing table to determine a forwarding interface and a next-hop IPv4 address. If a route is not found, send an ICMPv4 Destination Unreachable-Host Unreachable message to the source of the packet and discard the packet.
6. If the IPv4 MTU of the forwarding interface is less than the value of the Total Length field and the Don't Fragment (DF) flag is set to 0, perform IPv4 fragmentation. If the MTU of the forwarding interface is less than the value of the Total Length field and the DF flag is set to 1, send an ICMPv4 Destination Unreachable-Fragmentation Needed and DF Set message to the source of the packet and discard the packet.
7. Recalculate the new header checksum, and place its new value in the Header Checksum field.
8. Forward the packet by using the appropriate forwarding interface.

Note This critical router loop for IPv4 routers is a simplified list of items that an IPv4 router typically performs when forwarding. This list is not meant to imply any specific implementation nor an optimized order in which to process IPv4 packets for forwarding.

To forward a normal IPv6 packet, a router typically performs the following steps in its critical router loop:

1. Verify the value of the Version field. Although this step is not required by RFC 2460, performing it saves network bandwidth, because a packet containing an invalid version number is not propagated across the IPv6 internetwork only to be discarded by the destination node.

2. Decrement the value of the Hop Limit field. If its new value is less than 1, send an ICMPv6 Time Exceeded-Hop Limit Exceeded in Transit message to the source of the packet and discard the packet. If not, place the new value in the Hop Limit field.
3. Check the Next Header field for a value of 0. If it is 0, process the Hop-by-Hop Options header.
4. Use the value of the Destination Address field and the contents of the local routing table to determine a forwarding interface and a next-hop IPv6 address. If a route is not found, send an ICMPv6 Destination Unreachable-No Route To Destination message to the source of the packet and then discard the packet.
5. If the link MTU of the forwarding interface is less than 40 plus the value of the Payload Length field, send an ICMPv6 Packet Too Big message to the source of the packet and discard the packet.
6. Forward the packet by using the appropriate forwarding interface.

Note This critical router loop for IPv6 routers is a simplified list of items that an IPv6 router typically performs when forwarding. This list is not meant to imply any specific implementation nor an optimized order in which to process packets for forwarding.

As you can see, the process to forward an IPv6 packet is much simpler than for an IPv4 packet, as it does not have to verify and recalculate a header checksum, perform fragmentation, or process options not intended for the router.

IPv6 Extension Headers

The IPv4 header includes all options. Therefore, each intermediate router must check for their existence and process them when present. This can cause performance degradation in the forwarding of IPv4 packets. With IPv6, delivery and forwarding options are moved to extension headers. The only extension header that must be processed at each intermediate router is the Hop-by-Hop Options extension header. This increases IPv6 header processing speed and improves the performance of forwarding IPv6 packets.

RFC 2460 specifies that the following IPv6 extension headers must be supported by all IPv6 nodes:

- Hop-by-Hop Options header
- Destination Options header
- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header

With the exception of the Authentication header and Encapsulating Security Payload header, all the IPv6 extension headers just listed are defined in RFC 2460.

In a typical IPv6 packet, no extension headers are present. If special handling is required by either intermediate routers or the destination, the sending host adds one or more extension headers.

Each extension header must fall on a 64-bit (8-byte) boundary. Extension headers of a fixed size must be an integral multiple of 8 bytes. Extension headers of variable size contain a Header Extension Length field and must use padding as needed to ensure that their size is an integral multiple of 8 bytes.

The Next Header field in the IPv6 header and zero or more extension headers form a chain of pointers. Each pointer indicates the type of header that comes after the immediate header until the upper-layer protocol is ultimately identified. Figure 4-4 shows the chain of pointers formed by the Next Header field for various IPv6 packets.

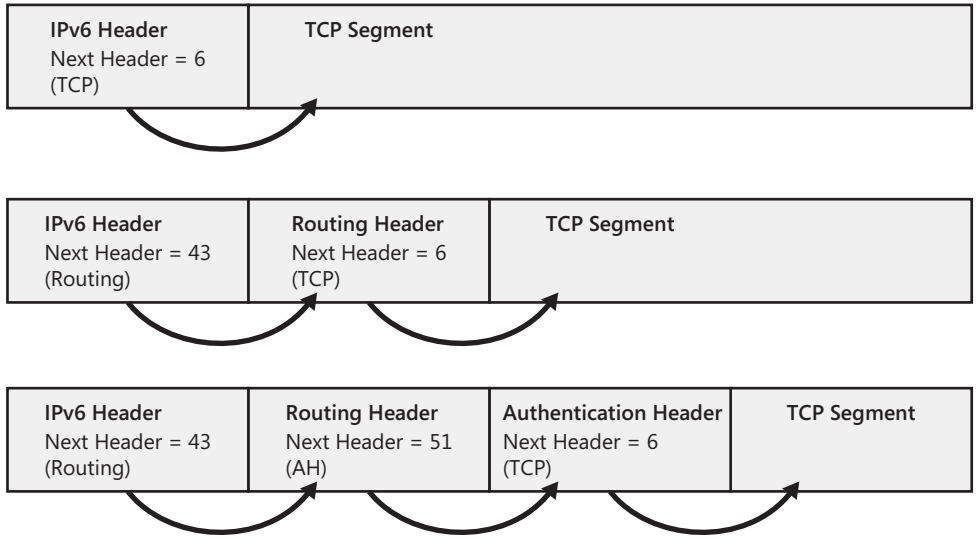

Figure 4-4 The chain of pointers formed by the Next Header field

Extension Headers Order

Extension headers are processed in the order in which they are present. Because the only extension header that is processed by every node on the path is the Hop-by-Hop Options header, it must be first. There are similar rules for other extension headers. In RFC 2460, it is recommended that extension headers be placed after the IPv6 header in the following order:

1. Hop-by-Hop Options header
2. Destination Options header (for intermediate destinations when the Routing header is present)

3. Routing header
4. Fragment header
5. Authentication header
6. Encapsulating Security Payload header
7. Destination Options header (for the final destination)

Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to specify delivery parameters at each hop on the path to the destination. It is identified by the value of 0 in the IPv6 header's Next Header field. Figure 4-5 shows the structure of the Hop-by-Hop Options header.

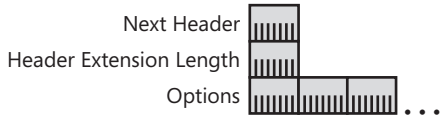

Figure 4-5 The structure of the Hop-by-Hop Options header

The Hop-by-Hop Options header consists of a Next Header field, a Header Extension Length field, and an Options field that contains one or more options. The value of the Header Extension Length field is the number of 8-byte blocks in the Hop-by-Hop Options extension header, not including the first 8 bytes. Therefore, for an 8-byte Hop-by-Hop Options header, the value of the Header Extension Length field is 0. Padding options are used to ensure 8-byte boundaries.

An IPv6 Router Optimization

The interpretation of the Header Extension Length field in the Hop-by-Hop Options header is another example of how the designers of IPv6 wanted to optimize processing of IPv6 packets at intermediate routers. For packets with a Hop-by-Hop Options header, one of the first operations is to determine the size of the header. If the Header Extension Length field were defined to be the number of 8-byte blocks in the header, its minimum value would be 1 (the minimum-sized Hop-by-Hop Options header is 8 bytes long). To ensure robustness in an IPv6 forwarding implementation, a field whose valid values begin at 1 has to be checked for the invalid value of 0 before additional processing can be done.

With the current definition of the Header Extension Length field, 0 is a valid value and no testing of invalid values needs to be done. The number of bytes in the Hop-by-Hop Options header is calculated from the following formula: (header extension length + 1) × 8.

An option is a set of fields that either describes a specific characteristic of the packet delivery or provides padding. Options are sent in the Hop-by-Hop Options header and Destination Options header (described later in this chapter). Each option is encoded in the type-length-value (TLV) format that is commonly used in TCP/IP protocols. Figure 4-6 shows the structure of an option.

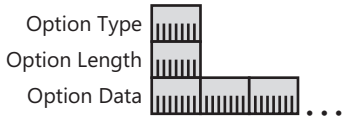

Figure 4-6 The structure of an option

The Option Type field both identifies the option and determines the way it is handled by the processing node. The Option Length field indicates the number of bytes in the option, not including the Option Type and Option Length fields. The option data is the specific data associated with the option.

An option might have an alignment requirement to ensure that specific fields within the option fall on desired boundaries. For example, it is easier to process an IPv6 address if it falls on an 8-byte boundary. Alignment requirements are expressed by using the notation $xn + y$, indicating that the option must begin at a byte boundary equal to an integral multiple of x bytes plus y bytes from the start of the header. For example, the alignment requirement $4n + 2$ indicates that the option must begin at a byte boundary of (an integral multiple of 4 bytes) + 2 bytes. In other words, the option must begin at the byte boundary of 6, 10, 14, and so on, relative to the start of the Hop-by-Hop Options or Destination Options headers. To accommodate alignment requirements, padding typically appears before an option and between each option when multiple options are present.

Option Type Field

Within the Option Type field, the two high-order bits indicate how the option is handled when the node processing the option does not recognize the option type. Table 4-3 lists the defined values of these two bits and their purpose.

Table 4-3 Values of the Two High-Order Bits in the Option Type Field

Value (Binary)	Action Taken
00	Skip the option.
01	Silently discard the packet.
10	Discard the packet, and send an ICMPv6 Parameter Problem message to the sender if the Destination Address field in the IPv6 header is a unicast or multicast address.
11	Discard the packet, and send an ICMPv6 Parameter Problem message to the sender if the Destination Address field in the IPv6 header is not a multicast address.

The third-highest-order bit of the Option Type indicates whether the option data can change (= 1) or not change (= 0) in the path to the destination.

Pad1 Option

The Pad1 option is defined in RFC 2460. It is used to insert a single byte of padding so that the Hop-by-Hop Options or Destination Options headers fall on 8-byte boundaries and to accommodate the alignment requirements of options. The Pad1 option has no alignment requirements. Figure 4-7 shows the Pad1 option.

Figure 4-7 The structure of the Pad1 option

The Pad1 option consists of a single byte; Option Type is set to 0, and it has no length or value fields. With Option Type set to 0, the option is skipped if not recognized and it is not allowed to change in transit.

PadN Option

The PadN option is defined in RFC 2460. It is used to insert two or more bytes of padding so that the Hop-by-Hop Options or Destination Options headers fall on 8-byte boundaries and to accommodate the alignment requirements of options. The PadN option has no alignment requirements. Figure 4-8 shows the PadN option.

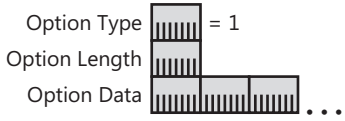

Figure 4-8 The structure of the PadN option

The PadN option consists of the Option Type field (set to 1), the Length field (set to the number of padding bytes present), and 0 or more bytes of padding. With the Option Type field set to 1, the option is skipped if not recognized and it is not allowed to change in transit.

Jumbo Payload Option

The Jumbo Payload option is defined in RFC 2675. It is used to indicate a payload size that is greater than 65,535 bytes. The Jumbo Payload option has the alignment requirement of 4n + 2. Figure 4-9 shows the Jumbo Payload option.

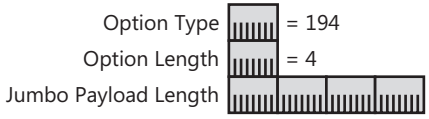

Figure 4-9 The structure of the Jumbo Payload option

With the Jumbo Payload option, the Payload Length field in the IPv6 header no longer indicates the size of the IPv6 packet payload. Instead, the Jumbo Payload Length field in the Jumbo Payload option indicates the size, in bytes, of the IPv6 packet payload. With a 32-bit Jumbo Payload Length field, payload sizes of up to 4,294,967,295 bytes can be indicated. An IPv6 packet with a payload size greater than 65,535 bytes is known as a *jumbogram*. With the Option Type field set to 194 (0xC2 hexadecimal, binary 11000010), the packet is discarded and an ICMPv6 Parameter Problem message is sent if the option is not recognized and the destination address is not a multicast address, and the option is not allowed to change in transit.

The IPv6 protocol in Windows Vista, Windows Server 2008, and Windows XP with Service Pack 2 supports incoming jumbograms at the IPv6 layer. However, there is no support in UDP or TCP for sending or receiving jumbograms.

Router Alert Option

The Router Alert option (Option Type 5) is defined in RFC 2711 and is used to indicate to a router that the contents of the packet require additional processing. The Router Alert option has the alignment requirement of 2n + 0. Figure 4-10 shows the structure of the Router Alert option.

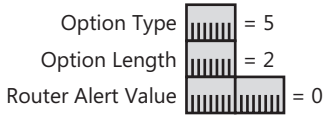

Figure 4-10 The structure of the Router Alert option

The Router Alert option is used for Multicast Listener Discovery (MLD) and the Resource ReSerVation Protocol (RSVP). With the Option Type field set to 5, the option is skipped if not recognized and it is not allowed to change in transit.

Network Monitor Capture

Here is an example of a Hop-by-Hop Options header as displayed by Network Monitor 3.1 (capture 04_02 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 32
  + Versions: IPv6, Internet Protocol, DSCP 0
    PayloadLength: 32 (0x20)
    NextProtocol: HOPOPT, IPv6 Hop-by-Hop Option, 0(0)
    HopLimit: 1 (0x1)
    SourceAddress: FE80:0:0:0:2B0:D0FF:FEE9:4143
    DestinationAddress: FF02:0:0:0:0:1:FFE9:4143
  - HopbyHopHeader:
     NextHeader: ICMPv6
     ExtHdrLen: 0(8 bytes)
   - OptionRouterAlert:
    - OptionType: Router Alert
       Action:     (00......) Skip over this option
```

```
       C:          (..0.....) Option Data does not change en-route
       OptionType: (...00101) Router Alert
      OptDataLen: 2 bytes
      Value: Datagram contains a Multicast Listener Discovery message, 0 (0x0)
   - OptionPadN:
    - OptionType: PadN
       Action:     (00......) Skip over this option
       C:          (..0.....) Option Data does not change en-route
       OptionType: (...00001) PadN
      OptDataLen: 0 bytes
      OptionData: 0 bytes
+ Icmpv6: Multicast Listener Report
```

Notice the use of the Router Alert option (option type 5) and the PadN option (option type 1) to pad the entire Hop-by-Hop Options header to 8 bytes (1-byte Next Header field + 1-byte Option Length field + 4-byte Router Alert option + 2-byte PadN option).

Destination Options Header

The Destination Options header is used to specify packet delivery parameters for either intermediate destinations or the final destination. This header is identified by the value of 60 in the previous header's Next Header field. The Destination Options header has the same structure as the Hop-by-Hop Options header, as shown in Figure 4-11.

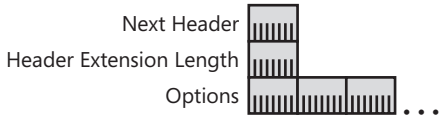

Figure 4-11 The structure of the Destination Options header

The Destination Options header is used in two ways:

1. If a Routing header is present, it specifies delivery or processing options at each intermediate destination. In this case, the Destination Options header occurs before the Routing header.
2. If no Routing header is present, or if this header occurs after the Routing header, this header specifies delivery or processing options at the final destination.

An example of a destination option is the Home Address destination option for Mobile IPv6.

Home Address Option

The Home Address destination option (Option Type 201) is defined in RFC 3775 and is used to indicate the home address of the mobile node. The home address is an address assigned to the mobile node when it is attached to the home link and through which the mobile node is always reachable, regardless of its location on an IPv6 network. For information about when the Home Address option is sent, see Appendix F, "Mobile IPv6." The Home Address option has the alignment requirement of 8n + 6. Figure 4-12 shows the structure of the Home Address option.

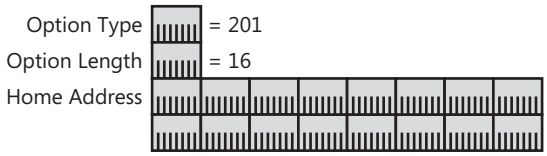

Figure 4-12 The structure of the Home Address option

The following list describes the fields in the Home Address option:

- **Option Type** With the Option Type field set to 201 (0xC9 hexadecimal, 11001001 binary), the packet is discarded and an ICMPv6 Parameter Problem message is sent if the option is not recognized and the destination address is not a multicast address, and the option is not allowed to change in transit.
- **Option Length** The Option Length field indicates the length of the option in bytes, not including the Option Type and Option Length fields. Because the only field past the Option Length field is the Home Address field to store an IPv6 address, the Option Length field is set to 16.
- **Home Address** The Home Address field indicates the home address of the mobile node. The size of this field is 128 bits.

For an example of the Home Address option in the Destination Options header, see the Network Monitor Capture 04_03 in the \NetworkMonitorCaptures folder on the companion CD-ROM.

Summary of Option Types

Table 4-4 lists the different option types for options in Hop-by-Hop Options and Destination Options headers.

Table 4-4 Option Types

Option Type	Option and Where It Is Used	Alignment Requirement
0	Pad1 option: Hop-by-Hop and Destination Options headers	None
1	PadN option: Hop-by-Hop and Destination Options headers	None
194 (0xC2)	Jumbo Payload option: Hop-by-Hop Options header	4n + 2
5	Router Alert option: Hop-by-Hop Options header	2n + 0
201 (0xC9)	Home Address option: Destination Options header	8n + 6

Routing Header

IPv4 defines strict source routing, in which each intermediate destination must be only one hop away, and loose source routing, in which each intermediate destination can be one or more hops away. IPv6 source nodes can use the Routing header to specify a source route, which is a list of intermediate destinations for the packet to travel to on its path to the final destination. The Routing header is identified by the value of 43 in the previous header's Next Header field. Figure 4-13 shows the structure of the Routing header.

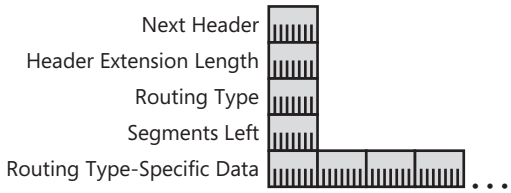

Figure 4-13 The structure of the Routing header

The Routing header consists of a Next Header field, a Header Extension Length field (defined in the same way as the Hop-by-Hop Options extension header), a Routing Type field, a Segments Left field that indicates the number of intermediate destinations that are still to be visited, and routing type-specific data.

RFC 2460 also defines Routing Type 0, used for loose source routing. Figure 4-14 shows the structure of the Routing Type 0 header.

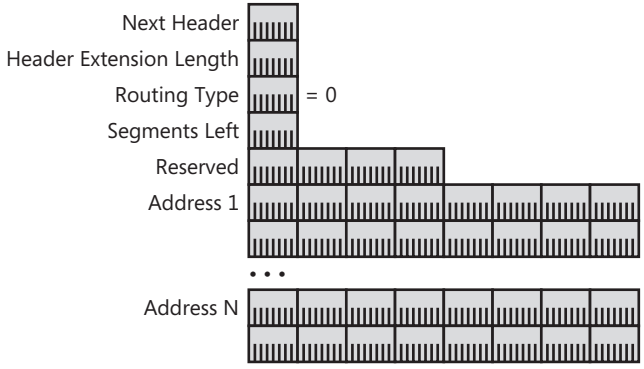

Figure 4-14 The structure of the Routing Type 0 header

For Routing Type 0, the routing type-specific data consists of a 32-bit Reserved field and a list of intermediate destination addresses, including the final destination address. When the packet is initially sent, the destination address is set to the first intermediate destination, and the routing type-specific data is the list of additional intermediate destinations and the final destination. The Segments Left field is set to the total number of addresses included in the routing type-specific data.

When the IPv6 packet reaches an intermediate destination, the Routing header is processed and the following actions are taken:

1. The current destination address and the address in the (*N* − Segments Left + 1) position in the list of addresses are swapped, where *N* is the total number of addresses in the Routing header.
2. The Segments Left field is decremented.
3. The packet is forwarded.

By the time the packet arrives at the final destination, the Segments Left field has been set to 0 and the list of intermediate addresses visited in the path to the destination is recorded in the Routing header.

IPv6 in Windows Vista will accept and process an incoming packet with a Routing Type 0 header. Because of security concerns, the Internet Engineering Task Force (IETF) is deprecating support for the Routing Type 0 header. IPv6 in Windows Server 2008 and Windows Vista Service Pack 1 will silently discard an incoming packet with a Routing Type 0 header.

Note Mobile IPv6 uses a Type 2 Routing header. For more information, see Appendix F, "Mobile IPv6."

Network Monitor Capture

Here is an example of the Routing header as displayed by Network Monitor 3.1 (capture 04_04 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 64
  + Versions: IPv6, Internet Protocol, DSCP 0
    PayloadLength: 64 (0x40)
    NextProtocol: IPv6 Routing header, 43(0x2b)
    HopLimit: 127 (0x7F)
    SourceAddress: FEC0:0:0:2:2B0:D0FF:FEE9:4143
    DestinationAddress: FEC0:0:0:2:260:97FF:FE02:6E8F
  - RoutingHeader:
     NextHeader: ICMPv6
     ExtHdrLen: 2(24 bytes)
     RoutingType: 0 (0x0)
     SegmentsLeft: 1 (0x1)
     Reserved: 0 (0x0)
     RouteAddress: FEC0:0:0:1:260:8FF:FE52:F9D8
+ Icmpv6: Echo request, ID = 0x0, Seq = 0x3d1a
```

In this simple example of the Routing header, an ICMPv6 Echo Request message is sent from the source FEC0::2:2B0:D0FF:FEE9:4143 to the destination FEC0::1:260:8FF:FE52:F9D8 using the intermediate destination of FEC0::2:260:97FF:FE02:6E8F.

Fragment Header

The Fragment header is used for IPv6 fragmentation and reassembly services. This header is identified by the value of 44 in the previous header's Next Header field. Figure 4-15 shows the structure of the Fragment header.

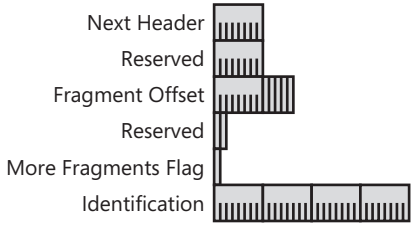

Figure 4-15 The structure of the Fragment header

The Fragment header includes a Next Header field, a 13-bit Fragment Offset field, a More Fragments flag, and a 32-bit Identification field. The Fragment Offset, More Fragments flag, and Identification fields are used in the same way as the corresponding fields in the IPv4 header. Because the use of the Fragment Offset field is defined for 8-byte fragment blocks, the Fragment header cannot be used for jumbograms. The maximum number that can be expressed with the 13-bit Fragment Offset field is 8191. Therefore, Fragment Offset can be used to indicate only a fragment data starting position of up to 8191 × 8, or 65,528.

In IPv6, only source nodes can fragment payloads. If the payload submitted by the upper-layer protocol is larger than the link or path MTU, IPv6 fragments the payload at the source and uses the Fragment header to provide reassembly information. An IPv6 router will never fragment an IPv6 packet being forwarded.

Because the IPv6 internetwork will not transparently fragment payloads, data sent from applications that do not have an awareness of the destination path MTU will not be able to sense when data needing fragmentation by the source is discarded by IPv6 routers. This can be a problem for unicast or multicast traffic sent as a UDP message or other types of message streams that do not use TCP.

Differences in Fragmentation Fields

There are some subtle differences between the fragmentation fields in IPv4 and IPv6. In IPv4, the fragmentation flags are the three high-order bits of the 16-bit quantity composed of the combination of the fragmentation flags and the Fragment Offset field. In IPv6, the bits used for fragmentation flags are the three low-order bits of the 16-bit quantity composed of the combination of the fragmentation flags and the Fragment Offset field. In IPv4, the Identification field is 16 bits rather than 32 bits in IPv6, and in IPv6 there is no Don't Fragment flag. Because IPv6 routers never perform fragmentation, the Don't Fragment flag is always set to 1 for all IPv6 packets and therefore does not need to be included.

IPv6 Fragmentation Process

When an IPv6 packet is fragmented, it is initially divided into unfragmentable and fragmentable parts:

- The unfragmentable part of the original IPv6 packet must be processed by intermediate nodes between the fragmenting node and the destination. This part consists of the IPv6 header, the Hop-by-Hop Options header, the Destination Options header for intermediate destinations, and the Routing header.
- The fragmentable part of the original IPv6 packet must be processed only at the final destination node. This part consists of the Authentication header, the Encapsulating Security Payload header, the Destination Options header for the final destination, and the upper-layer PDU.

Next, the IPv6 fragment packets are formed. Each fragment packet consists of the unfragmentable part, a fragment header, and a portion of the fragmentable part. Figure 4-16 shows the IPv6 fragmentation process for a packet fragmented into three fragments.

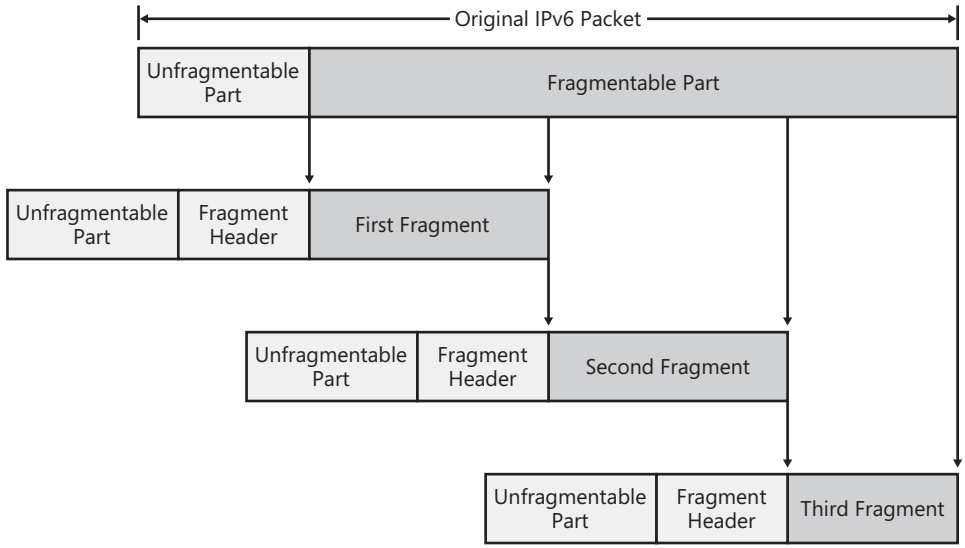

Figure 4-16 The IPv6 fragmentation process

In each fragment, the Next Header field in the Fragment header indicates the first header or the upper-layer protocol in the original fragmentable part. The Fragment Offset field in the Fragment header indicates the offset, in 8-byte units known as *fragment blocks*, of this fragment relative to the original payload. The More Fragments flag is set on all fragment packets except the last fragment packet. All fragment packets created from the same IPv6 packet must contain the same Identification field value.

Fragmentation of IPv6 packets can occur when the upper-layer protocol of the sending host submits a packet to IPv6 that is larger than the path MTU to the destination. Examples of IPv6 fragmentation are when a UDP application that is not aware of a path MTU sends large packets to a destination, or when a TCP application sends a packet before it is made aware of a path MTU update that lowers the path MTU. In this latter case, IPv6 is aware of the new path MTU, but TCP is not. TCP submits the TCP segment by using the old, larger value of the path MTU, and IPv6 fragments the TCP segment to fit the new, lower path MTU value. Once TCP is made aware of the new path MTU, subsequent TCP segments are not fragmented.

IPv6 packets sent to IPv4 destinations that undergo IPv6-to-IPv4 header translation might receive a path MTU update of less than 1280. In this case, the sending host sends IPv6 packets with a Fragment header in which the Fragment Offset field is set to 0 and the More Fragments flag is not set, and with a smaller payload size of 1272 bytes. The Fragment header is included so that the IPv6-to-IPv4 translator can use the Identification field in the Fragment header to perform IPv4 fragmentation to reach the IPv4 destination.

Network Monitor Capture

Here is an example of a Fragment header as displayed by Network Monitor 3.1 (frame 3 of capture 04_05 in the \NetworkMonitorCaptures folder on the companion CD-ROM):

```
  Frame:
+ Ethernet: Etype = IPv6
- Ipv6: Next Protocol = ICMPv6, Payload Length = 1456
  + Versions: IPv6, Internet Protocol, DSCP 0
    PayloadLength: 1456 (0x5B0)
    NextProtocol: IPv6 Fragment header, 44(0x2c)
    HopLimit: 128 (0x80)
    SourceAddress: FE80:0:0:0:210:5AFF:FEAA:20A2
    DestinationAddress: FE80:0:0:0:250:DAFF:FED8:C153
  - FragmentHeader:
     NextHeader: ICMPv6
     Reserved: 0 (0x0)
   - FragmentInfor:
      FragmentOffset: 2896(0XB50)
      Reserved: (.............00.)
      M:        (...............1) More fragments
      Identification: 5 (0x5)
     FragmentData: Binary Large Object (1448 Bytes)
```

This is a fragment of a payload that uses the identification number of 5 and starts in byte position 2896 relative to the fragmentable portion of the original IPv6 payload.

IPv6 Reassembly Process

The fragment packets are forwarded by the intermediate IPv6 router or routers to the destination IPv6 address. The fragment packets can take different paths to the destination and arrive in a different order from which they were sent. To reassemble the fragment packets into the original payload, IPv6 uses the Source Address and Destination Address fields in the IPv6

header and the Identification field in the Fragment header to group the fragments. Figure 4-17 shows the IPv6 reassembly process.

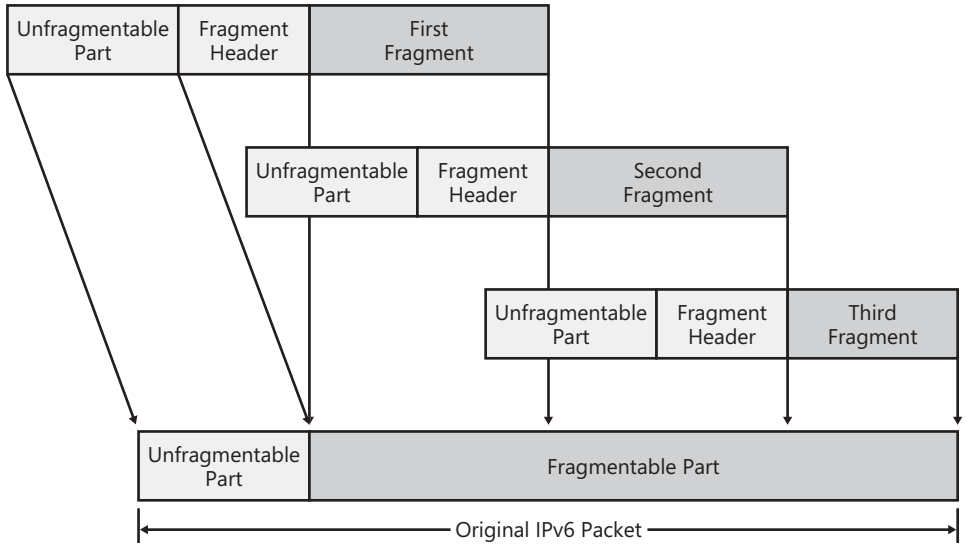

Figure 4-17 The IPv6 reassembly process

After all the fragments arrive, the original payload length is calculated and the Payload Length field in the IPv6 header for the reassembled packet is updated. Additionally, the Next Header field of the last header of the unfragmentable part is set to the Next Header field of the Fragment header of the first fragment.

RFC 2460 recommends a reassembly time of 60 seconds before abandoning reassembly and discarding the partially reassembled packet. If the first fragment has arrived and reassembly has not completed, the reassembling host sends an ICMPv6 Time Exceeded-Fragment Reassembly Time Exceeded message to the source of the fragment.

Authentication Header

The Authentication header provides data authentication (verification of the node that sent the packet), data integrity (verification that the data was not modified in transit), and antireplay protection (assurance that captured packets cannot be retransmitted and accepted as valid data) for the IPv6 packet including the fields in the IPv6 header that do not change in transit across an IPv6 internetwork. The Authentication header, described in RFC 2402, is part of the security architecture for IP, as defined in RFC 2401. The Authentication header is identified by the value of 51 in the previous header's Next Header field. Figure 4-18 shows the structure of the Authentication header.

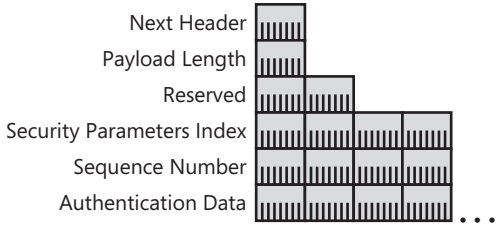

Figure 4-18 The structure of the Authentication header

The Authentication header contains a Next Header field, a Payload Length field (the number of 4-byte blocks in the Authentication header, not counting the first two), a Reserved field, a Security Parameters Index (SPI) field that helps identify a specific IP Security (IPsec) security association (SA), a Sequence Number field that provides antireplay protection, and an Authentication Data field that contains an integrity value check (ICV). The ICV provides data authentication and data integrity.

The Authentication header does not provide data confidentiality services for the upper-layer PDU by encrypting the data so that it cannot be viewed without the encryption key. To obtain data authentication and data integrity for the entire IPv6 packet and data confidentiality for the upper-layer PDU, you can use both the Authentication header and the Encapsulating Security Payload header and trailer.

Encapsulating Security Payload Header and Trailer

The Encapsulating Security Payload (ESP) header and trailer, described in RFC 2406, provide data confidentiality, data authentication, data integrity, and replay protection services to the encapsulated payload. The ESP header provides no security services for the IPv6 header or extension headers that occur before the ESP header. The ESP header and trailer are identified by the value of 50 in the previous header's Next Header field. Figure 4-19 shows the structure of the ESP header and trailer.

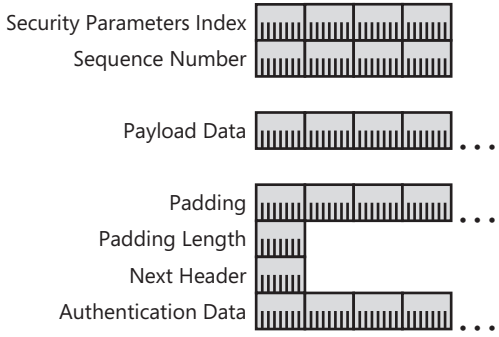

Figure 4-19 The structure of the Encapsulating Security Payload header and trailer

The ESP header contains an SPI field that helps identify the IPsec SA, and a Sequence Number field that provides antireplay protection. The ESP trailer contains the Padding, Padding Length, Next Header, and Authentication Data fields. The Padding field is used to ensure 4-byte boundaries for the ESP payload and appropriate data-block boundaries for encryption algorithms. The Padding Length field indicates the size of the Padding field in bytes. The Authentication Data field contains the ICV.

Details about how the ESP header and trailer provide data confidentiality, authentication, and integrity through cryptographic techniques are beyond the scope of this book.

IPv6 MTU

IPv6 requires that the link layer support a minimum MTU size of 1280 bytes. Link layers that do not support this MTU size must provide a link-layer fragmentation and reassembly scheme that is transparent to IPv6. For link layers that can support a configurable MTU size, RFC 2460 recommends that they be configured with an MTU size of at least 1500 bytes (the IPv6 MTU for Ethernet II encapsulation). An example of a configurable MTU is the Maximum Receive Unit (MRU) of a Point-to-Point Protocol (PPP) link.

Like IPv4, IPv6 provides a Path MTU Discovery process that uses the ICMPv6 Packet Too Big message described in the "Path MTU Discovery" section of Chapter 5, "ICMPv6." Path MTU Discovery allows the transmission of IPv6 packets that are larger than 1280 bytes.

IPv6 source hosts can fragment payloads of upper-layer protocols that are larger than the path MTU by using the process and Fragment header previously described. However, the use of IPv6 fragmentation is highly discouraged. An IPv6 node must be able to reassemble a fragmented packet that is at least 1500 bytes in size.

Table 4-5 lists commonly used local area network (LAN) and wide area network (WAN) technologies and their defined IPv6 MTUs.

Table 4-5 IPv6 MTUs for Common LAN and WAN Technologies

LAN or WAN Technology	IPv6 MTU
Ethernet (Ethernet II encapsulation)	1500
Ethernet (IEEE 802.3 SubNetwork Access Protocol [SNAP] encapsulation)	1492
IEEE 802.11	2312
Token Ring	Varies
Fiber Distributed Data Interface (FDDI)	4352
Attached Resource Computer Network (ARCNet)	9072
PPP	1500
X.25	1280
Frame Relay	1592
Asynchronous Transfer Mode (ATM) (Null or SNAP encapsulation)	9180

For more information about LAN and WAN encapsulations for IPv6 packets, see Appendix A, "Link-Layer Support for IPv6."

Upper-Layer Checksums

The current implementation of TCP, UDP, and ICMP for IPv4 incorporates into their checksum calculation a pseudo-header that includes both the IPv4 Source Address and Destination Address fields. This checksum calculation must be modified for TCP, UDP, and ICMPv6 traffic sent over IPv6 to include IPv6 addresses. Figure 4-20 shows the structure of the new IPv6 pseudo-header that must be used by TCP, UDP, and ICMPv6 checksum calculations. IPv6 uses the same algorithm as IPv4 for computing the checksum value.

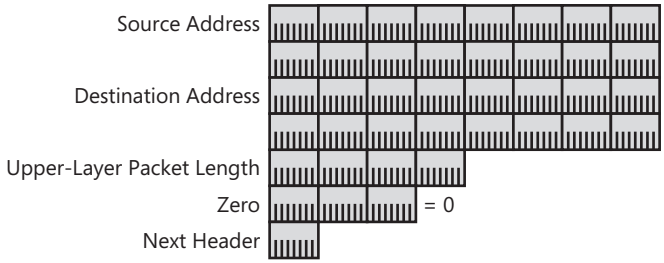

Figure 4-20 The structure of the new IPv6 pseudo-header

The IPv6 pseudo-header includes the Source Address field, the Destination Address field, an Upper Layer Packet Length field that indicates the length of the upper-layer PDU, and a Next Header field that indicates the upper-layer protocol for which the checksum is being calculated.

References

The following references were cited in this chapter:

- RFC 791 — "Internet Protocol"
- RFC 1812 — "Requirements for IP Version 4 Routers"
- RFC 2401 — "Security Architecture for the Internet Protocol"
- RFC 2402 — "IP Authentication Header"
- RFC 2406 — "IP Encapsulating Security Payload (ESP)"
- RFC 2460 — "Internet Protocol, Version 6 (IPv6)"
- RFC 2474 — "Definition of the Differentiated Services Field (DS Field)"
- RFC 2675 — "IPv6 Jumbograms"
- RFC 2711 — "IPv6 Router Alert Option"

- RFC 3168 — "The Addition of Explicit Congestion Notification (ECN) to IP"
- RFC 3697 — "IPv6 Flow Label Specification"
- RFC 3775 — "Mobility Support in IPv6"

You can obtain these RFCs from the \RFCs_and_Drafts folder on the companion CD-ROM or from *http://www.ietf.org/rfc.html*.

Testing for Understanding

To test your understanding of the IPv6 header, answer the following questions. See Appendix D, "Testing for Understanding Answers," to check your answers.

1. Why does the IPv6 header not include a checksum?
2. What is the IPv6 equivalent to the IHL field in the IPv4 header?
3. How does the combination of the Traffic Class and Flow Label fields provide better support for prioritized traffic delivery?
4. Which extension headers are fragmentable and why? Which extension headers are not fragmentable and why?
5. Describe a situation that results in an IPv6 packet that contains a Fragment Header in which the Fragment Offset field is set to 0 and the More Fragments flag is not set to 1.
6. Describe how the new upper-layer checksum calculation affects transport layer protocols such as TCP and UDP.
7. If the minimum MTU for IPv6 packets is 1280 bytes, how are 1280-byte packets sent on a link that supports only 512-byte frames?