### 1.1 TCP/IP architectural model

The TCP/IP protocol suite is so named for two of its most important protocols:

Transmission Control Protocol (TCP) and Internet Protocol (IP). A less used

name for it is the Internet Protocol Suite, which is the phrase used in official Internet standards documents. In this book, we use the more common, shorter term, TCP/IP, to refer to the entire protocol suite.

#### 1.1.1 Internetworking

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

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

The Internet consists of the following groups of networks:

Backbones: Large networks that exist primarily to interconnect other networks. Also known as network access points (NAPs) or Internet Exchange Points (IXPs). Currently, the backbones consist of commercial entities.

Regional networks connecting, for example, universities and colleges.

Commercial networks providing access to the backbones to subscribers, and networks owned by commercial organizations for internal use that also have connections to the Internet.

Local networks, such as campus-wide university networks.

In most cases, networks are limited in size by the number of users that can belong to the network, by the maximum geographical distance that the network can span, or by the applicability of the network to certain environments. For example, an Ethernet network is inherently limited in terms of geographical size. Therefore, the ability to interconnect a large number of networks in some

hierarchical and organized fashion enables the communication of any two hosts belonging to this internetwork.

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

rconnected by a

Two networks inte

rou

ter

equals

Internet A

Router

R

One

Virtual

Network

Netw

ork 1

Network 2

Router

R

Net

work 3

Netw

ork 1

Network 2

Router

R

Multiple network

s interconnected by routers

(

also seen as 1 virtual network, an Internet

)

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

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

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

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

IP address = <network number><host number>

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

#### 1.1.2 The TCP/IP protocol layers

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

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

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

Applications

Transport

Internetwork

Network Interface

and

Hardware

Applications

TCP/UDP

ICMP

IP

ARP/RARP

Network Interface

and Hardware

.......

.......

.......

.......

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

|  |  |
| --- | --- |
| **Application layer** | The application layer is provided by the program that uses TCP/IP for communication. An application is a user process cooperating with another process usually on a different host (there is also a benefit to application communication within a single host). Examples of applications include Telnet and the File Transfer Protocol (FTP). The interface between the application and transport layers is defined by port numbers and sockets, which we describe in more detail in 4.1, “Ports and sockets” on page 144. |
| **Transport layer** | The transport layer provides the end-to-end data transfer by delivering data from an application to its remote peer. Multiple applications can be supported simultaneously. The most-used transport layer protocol is the Transmission Control Protocol (TCP), which provides connection-oriented reliable data delivery, duplicate data suppression, congestion control, and flow control. We discuss this in more detail in 4.3, “Transmission Control Protocol (TCP)” on    page 149. |

Another transport layer protocol is the User Datagram Protocol (see 4.2, “User Datagram Protocol (UDP)” on page 146). It provides connectionless, unreliable, best-effort service. As a result, applications using UDP as the transport protocol have to provide their own end-to-end integrity, flow control, and congestion control, if desired. Usually, UDP is used by applications that need a fast transport mechanism and can tolerate the loss of some data.

|  |  |
| --- | --- |
| **Internetwork layer** | The internetwork layer, also called the *internet layer* or the *network layer*, provides the “virtual network” image of an internet (this layer shields the higher levels from the physical network architecture below it). Internet Protocol (IP) is the most important protocol in this layer. It is a connectionless protocol that does not assume reliability from lower layers. IP does *not* provide reliability, flow control, or error recovery. These functions must be provided at a higher level.  IP provides a routing function that attempts to deliver transmitted messages to their destination. We discuss IP in detail in Chapter 3, “Internetworking protocols” on page 67. A message unit in an IP network is called an *IP datagram*. This is the basic unit of information transmitted across TCP/IP networks. Other internetwork-layer protocols are IP, ICMP, IGMP, ARP, and RARP. |
| **Network interface layer** The network interface layer, also called the *link layer* or the *data-link layer*, is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer. Examples are IEEE 802.2, X.25 (which is reliable in itself), ATM, FDDI, and even SNA. We discuss some physical networks and interfaces in Chapter 2, “Network interfaces” on page 29. | |

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

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

Applications

Transport

Internetwork

Network Interface

and Hardware

SMTP, Telnet, FTP, Gopher...

UDP

TCP

IP

ICMP

ARPRARP

Ethernet, Token-Ring, FDDI, X.25, Wireless, Async, ATM,

SNA...

*Figure 1-3 Detailed architectural model*

#### 1.1.3 TCP/IP applications

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

All application protocols have some characteristics in common:

They can be user-written applications or applications standardized and shipped with the TCP/IP product. Indeed, the TCP/IP protocol suite includes application protocols such as:

* Telnet for interactive terminal access to remote internet hosts
* File Transfer Protocol (FTP) for high-speed disk-to-disk file transfers
* Simple Mail Transfer Protocol (SMTP) as an internet mailing system

These are some of the most widely implemented application protocols, but many others exist. Each particular TCP/IP implementation will include a lesser or greater set of application protocols.

They use either UDP or TCP as a transport mechanism. Remember that UDP is unreliable and offers no flow-control, so in this case, the application has to provide its own error recovery, flow control, and congestion control functionality. It is often easier to build applications on top of TCP because it is a reliable stream, connection-oriented, congestion-friendly, flow control-enabled protocol. As a result, most application protocols will use TCP, but there are applications built on UDP to achieve better performance through increased protocol efficiencies.

Most applications use the client/server model of interaction.

##### The client/server model

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

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | |  | | |  | |  | | |  | |
|  | Cli  A | ent |  |  |  | Client  B | |  | ..... |  | Server | |  |  |
|  | TC | P/IP |  | TCP/IP | |  | TCP/IP | |
|  |  |  |  |  |  |  |  |  |
|  |  | |  |  | |  |  | |
|  | | Internet Network | | | | | | | |  | | | |

*Figure 1-4 The client/server model of applications*

Most servers wait for requests at a *well-known port* so that their clients know to which port (and in turn, which application) they must direct their requests. The client typically uses an arbitrary port called an *ephemeral port* for its communication. Clients that want to communicate with a server that does not use a well-known port must have another mechanism for learning to which port they must address their requests. This mechanism might employ a registration service such as portmap, which does use a well-known port. For detailed information about TCP/IP application protocols, refer to Part 2,

“TCP/IP application protocols” on page 405.

##### Bridges, routers, and gateways

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

|  |  |
| --- | --- |
| **Bridge** | Interconnects LAN segments at the network interface layer level and forwards frames between them. A bridge performs the function of a MAC relay, and is independent of any higher layer protocol (including the logical link protocol). It provides MAC layer protocol conversion, if required.  A bridge is said to be *transparent* to IP. That is, when an IP host sends an IP datagram to another host on a network connected by a bridge, it sends the datagram directly to the host and the datagram “crosses” the bridge without the sending IP host being aware of it. |
| **Router** | Interconnects networks at the internetwork layer level and  routes packets between them. The router must understand the addressing structure associated with the networking protocols it supports and take decisions on whether, or how, to forward packets. Routers are able to select the best transmission paths and optimal packet sizes. The basic routing function is implemented in the IP layer of the TCP/IP protocol stack, so any host or workstation running TCP/IP over more than one interface could, in theory and also with most of today's TCP/IP implementations, forward IP datagrams. However, dedicated routers provide much more sophisticated routing than the minimum functions implemented by IP. |

Because IP provides this basic routing function, the term

“IP router,” is often used. Other, older terms for router are “IP gateway,” “Internet gateway,” and “gateway.” The term *gateway* is now normally used for connections at a higher layer than the internetwork layer.

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

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

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

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

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

### 1.2 The roots of the Internet

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

A number of networks were installed in the late 1960s and 1970s, when network design was the “state of the art” topic of computer research and sophisticated implementers. It resulted in multiple networking models such as packet-switching

technology, collision-detection local area networks, hierarchical networks, and many other excellent communications technologies.

The result of all this great know-how was that any group of users could find a physical network and an architectural model suitable for their specific needs. This ranges from inexpensive asynchronous lines with no other error recovery than a bit-per-bit parity function, through full-function wide area networks (public or private) with reliable protocols such as public packet-switching networks or private SNA networks, to high-speed but limited-distance local area networks.

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

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

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

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

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

#### 1.2.1 ARPANET

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

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

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

#### 1.2.2 NSFNET

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

The backbone: A network that connects separately administered and operated mid-level networks and NSF-funded supercomputer centers. The backbone also has transcontinental links to other networks such as EBONE, the European IP backbone network.

Mid-level networks: Three kinds of networks (regional, discipline-based, and supercomputer consortium networks).

Campus networks: Whether academic or commercial, connected to the mid-level networks.

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

First backbone: Originally established by the NSF as a communications network for researchers and scientists to access the NSF supercomputers, the first NSFNET backbone used six DEC LSI/11 microcomputers as packet switches, interconnected by 56 Kbps leased lines. A primary interconnection between the NSFNET backbone and the ARPANET existed at Carnegie Mellon, which allowed routing of datagrams between users connected to each of those networks.

Second backbone: The need for a new backbone appeared in 1987, when the first one became overloaded within a few months (estimated growth at that time was 100% per year). The NSF and MERIT, Inc., a computer network consortium of eight state-supported universities in Michigan, agreed to develop and manage a new, higher-speed backbone with greater transmission and switching capacities. To manage it, they defined the *Information Services* (IS), which is comprised of an Information Center and a Technical Support Group. The Information Center is responsible for information dissemination, information resource management, and electronic communication. The Technical Support Group provides support directly to the field. The purpose of this is to provide an integrated information system with easy-to-use-and-manage interfaces accessible from any point in the network supported by a full set of training services.

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

UNIX) loosely coupled by two IBM token-ring networks (for redundancy). One

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

* Dynamic alternate routing
* Dynamic bandwidth allocation

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

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

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

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

#### 1.2.3 Commercial use of the Internet

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

One important initiative to consider is that of the *Acceptable Use Policy* (AUP). The first of these policies was introduced in 1992 and applies to the use of NSFNET. At the heart of this AUP is a commitment “to support open research and education.” Under “Unacceptable Uses” is a prohibition of “use for for-profit activities,” unless covered by the General Principle or as a specifically acceptable use. However, in spite of this apparently restrictive stance, the NSFNET was increasingly used for a broad range of activities, including many of a commercial nature, before reverting to its original objectives in 1995.

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

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

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

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

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

[http://www.ep.net](http://www.ep.net/)

Similar to CIX in the United States, European Internet providers formed the RIPE

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

#### 1.2.4 Internet2

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

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

##### Mission

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

Internet2 has the following goals:

Demonstrate new applications that can dramatically enhance researchers’ ability to collaborate and conduct experiments.

Demonstrate enhanced delivery of education and other services (for instance, health care, environmental monitoring, and so on) by taking advantage of *virtual proximity* created by an advanced communications infrastructure.

Support development and adoption of advanced applications by providing middleware and development tools.

Facilitate development, deployment, and operation of an affordable communications infrastructure, capable of supporting differentiated quality of service (QoS) based on application requirements of the research and education community.

Promote experimentation with the next generation of communications technologies.

Coordinate adoption of agreed working standards and common practices among participating institutions to ensure end-to-end quality of service and interoperability.

Catalyze partnerships with governmental and private sector organizations.

Encourage transfer of technology from Internet2 to the rest of the Internet.

Study the impact of new infrastructure, services, and applications on higher education and the Internet community in general.

##### Internet2 participants

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

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

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

For more information about Internet2, see their Web page at: [http://www.internet2.edu](http://www.internet2.edu/)

#### 1.2.5 The Open Systems Interconnection (OSI) Reference Model

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

Application

Presentation

Session

Transport

Network

Data Link

Physical

Application

Presentation

Session

Transport

Network

Data Link

Physical

*Figure 1-5 The OSI Reference Model*

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

The seven layers are:

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

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

### 1.3 TCP/IP standards

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

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

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

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

The overall goals of the Internet Standards Process are:

Technical excellence

Prior implementation and testing Clear, concise, and easily understood documentation

Openness and fairness

Timeliness

The process of standardization is summarized as follows:

In order to have a new specification approved as a standard, applicants have to submit that specification to the IESG where it will be discussed and reviewed for technical merit and feasibility and also published as an Internet draft document. This should take no shorter than two weeks and no longer than six months.

After the IESG reaches a positive conclusion, it issues a last-call notification to allow the specification to be reviewed by the whole Internet community.

After the final approval by the IESG, an Internet draft is recommended to the Internet Engineering Taskforce (IETF), another subsidiary of the IAB, for inclusion into the Standards Track and for publication as a Request for Comments (see 1.3.1, “Request for Comments (RFC)” on page 22).

Once published as an RFC, a contribution may advance in status as described in 1.3.2, “Internet standards” on page 24. It may also be revised over time or phased out when better solutions are found.

If the IESG does not approve of a new specification after, or if a document has remained unchanged within, six months of submission, it will be removed from the Internet drafts directory.

#### 1.3.1 Request for Comments (RFC)

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

After an RFC has been published, all revisions and replacements are published as new RFCs. A new RFC that revises or replaces an existing RFC is said to

“update” or to “obsolete” that RFC. The existing RFC is said to be “updated by” or

“obsoleted by” the new one. For example RFC 1542, which describes the BOOTP protocol, is a “second edition,” being a revision of RFC 1532 and an amendment to RFC 951. RFC 1542 is therefore labelled like this: “Obsoletes RFC 1532; Updates RFC 951." Consequently, there is never any confusion over whether two people are referring to different versions of an RFC, because there is never more than one current version.

Some RFCs are described as *information documents*, while others describe

Internet protocols. The Internet Architecture Board (IAB) maintains a list of the RFCs that describe the protocol suite. Each of these is assigned a *state* and a *status*.

An Internet protocol can have one of the following states:

|  |  |
| --- | --- |
| **Standard** | The IAB has established this as an official protocol for the Internet. These are separated into two groups:  IP protocol and above, protocols that apply to the whole Internet  Network-specific protocols, generally specifications of how to do IP on particular types of networks |
| **Draft standard** | The IAB is actively considering this protocol as a possible standard protocol. Substantial and widespread testing and comments are desired. Submit comments and test results to the IAB. There is a possibility that changes will be made in a draft protocol before it becomes a standard. |
| **Proposed standard** | These are protocol proposals that might be considered by the IAB for standardization in the future. Implementations and testing by several groups are desirable. Revision of the protocol is likely. |
| **Experimental** | A system should not implement an experimental protocol unless it is participating in the experiment and has coordinated its use of the protocol with the developer of the protocol. |
| **Informational** | Protocols developed by other standard organizations, or vendors, or that are for other reasons outside the purview of the IAB may be published as RFCs for the convenience of the Internet community as informational protocols. Such protocols might, in some cases, also be recommended for use on the Internet by the IAB. |
| **Historic** | These are protocols that are unlikely to ever become standards in the Internet either because they have been  superseded by later developments or due to lack of interest. |

Protocol status can be any of the following:

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

#### 1.3.2 Internet standards

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

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

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

The following Internet standards are of particular importance:

STD 1 – Internet Official Protocol Standards

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

STD 2 – Assigned Internet Numbers

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

STD 3 – Host Requirements

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

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

STD 4 – Router Requirements

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

##### For Your Information (FYI)

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

##### Obtaining RFCs

RFC and ID documents are available publicly and online and best obtained from the IETF Web site: [http://www.ietf.org](http://www.ietf.org/)

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

### 1.4 Future of the Internet

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

#### 1.4.1 Multimedia applications

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

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

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

#### 1.4.2 Commercial use

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

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

#### 1.4.3 The wireless Internet

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

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

### 1.5 RFCs relevant to this chapter

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

[RFC 2026 – The Internet Standards Process -- Revision 3 (October 1996)](ftp://ftp.rfc-editor.org/in-notes/rfc2026.txt)

[RFC 2223 – Instructions to RFC (October 1997)](ftp://ftp.rfc-editor.org/in-notes/rfc2223.txt)

[RFC 2900 – Internet Official Protocol Standards (August 2001)](ftp://ftp.rfc-editor.org/in-notes/rfc2900.txt)

[RFC 3232 – Assigned Numbers: RFC 1700 is Replaced by an On-line](ftp://ftp.rfc-editor.org/in-notes/rfc3232.txt)

[Database (January 2002)](ftp://ftp.rfc-editor.org/in-notes/rfc3232.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **3** | |

## Chapter 3. Internetworking protocols

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

Internet Protocol (IP)

Internet Control Message Protocol (ICMP)

Address Resolution Protocol (ARP)

Dynamic Host Configuration Protocol (DHCP)

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

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

### 3.1 Internet Protocol (IP)

IP is a standard protocol with STD number 5. The standard also includes ICMP

(see 3.2, “Internet Control Message Protocol (ICMP)” on page 109) and IGMP (see 3.3, “Internet Group Management Protocol (IGMP)” on page 119). IP has a status of required.

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

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

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

#### 3.1.1 IP addressing

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

##### The IP address

IP addressing standards are described in RFC 1166. To identify a host on the

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

IP address = <network number><host number>

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

American Registry for Internet Numbers (ARIN): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers for North America, South America, the Caribbean, and sub-Saharan Africa.

Reseaux IP Europeans (RIPE): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers for Europe, Middle East, and parts of Africa.

Asia Pacific Network Information Centre (APNIC): This registry is responsible for the administration and registration of Internet Protocol (IP) numbers within the Asia Pacific region.

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

The binary format of the IP address 128.2.7.9 is:

10000000 00000010 00000111 00001001

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

##### Class-based IP addresses

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class A  Class B  Class C  Class D  Class E | 1 2 3  01 8 6 4 1 | | | | | | | | | | |
|  | |  | | | | |  | | |  |
| 0 |  | netID | | | | | hostID | | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | |  | | | | |  | |  |
|  | 10 | | netID | | | | | hostID | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | |  | | | | |  |  |
|  | 110 | | | netID | | | | | hostID |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | | |  | | | | |  |
|  | 1110 | | | | multicast | | | | |  |
|  |  | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | |  | | | |  |
|  | 11110 | | | | | future/experimental use | | | |  |
|  |  | | | | | | | | |

*Figure 3-1 IP: Assigned classes of IP addresses*

Where:

|  |  |
| --- | --- |
| **Class A addresses** | These addresses use 7 bits for the <network> and 24 bits for the <host> portion of the IP address. This allows for 27-2 (126) networks each with 224-2 (16777214) hosts—a total of more than 2 billion addresses. |
| **Class B addresses** | These addresses use 14 bits for the <network> and 16 bits for the <host> portion of the IP address. This allows for 214-2 (16382) networks each with 216-2 (65534) hosts—a total of more than 1 billion addresses. |
| **Class C addresses** | These addresses use 21 bits for the <network> and 8 bits for the <host> portion of the IP address. That allows for 221-2 (2097150) networks each with 28-2 (254) hosts—a total of more than half a billion addresses. |
| **Class D addresses** | These addresses are reserved for multicasting (a sort of |
| **Class E addresses** | broadcasting, but in a limited area, and only to hosts using the same Class D address).  These addresses are reserved for future or experimental use. |

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

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

##### Reserved IP addresses

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

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

All bits 1: An address with all bits one is interpreted as *all* networks or *all* hosts. For example, the following means all hosts on network 128.2 (Class B address):

128.2.255.255

This is called a directed broadcast address because it contains both a valid <network address> and a broadcast <host address>.

Loopback: The Class A network 127.0.0.0 is defined as the loopback network. Addresses from that network are assigned to interfaces that process data within the local system. These loopback interfaces do not access a physical network.

##### Special use IP addresses

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

*Table 3-1 Special use IP addresses*

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

#### 3.1.2 IP subnets

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

A new type of physical network is installed at a location.

Growth of the number of hosts requires splitting the local network into two or more separate networks.

Growing distances require splitting a network into smaller networks, with gateways between them.

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

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

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

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

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

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

The first octet is the subnet number; the second octet is the host number. This gives 28-2 (254) possible subnets, each having up to 28-2 (254) hosts. Recall that we subtract two from the possibilities to account for the all ones and all zeros cases. The subnet mask is 255.255.255.0.

The first 12 bits are used for the subnet number and the last four for the host number. This gives 212-2 (4094) possible subnets but only 24-2 (14) hosts per subnet. The subnet mask is 255.255.255.240.

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

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

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

##### Types of subnetting

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

###### Static subnetting

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

###### Variable length subnetting

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

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

###### Mixing static and variable length subnetting

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

##### Static subnetting example

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

0

Class A

netID

hostID

1 2 3

01

8 6 4

1

*Figure 3-2 IP: Class A address without subnets* Use the IP address shown in Figure 3-3.

|  |
| --- |
| 00001001 01000011 00100110 00000001 a 32-bit address  9 67 38 1 decimal notation (9.67.38.1) |

*Figure 3-3 IP address*

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

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

1 2 3

01

8 6 4

1

Class A

Subnet

subnet number

0

host

ID

netID

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

A bit mask, known as the subnet mask, is used to identify which bits of the original host address field indicate the subnet number. In the previous example, the subnet mask is 255.255.255.192 (or 11111111 11111111 11111111

11000000 in bit notation). Note that, by convention, the <network address> is included in the mask as well.

Because of the all bits 0 and all bits 1 restrictions, this defines 218-2 (from 1 to 262143) valid subnets. This split provides 262142 subnets each with a maximum of 26-2 (62) hosts.

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

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

00001001 01000011 00100110 00000001 = 9.67.38.1 (Class A address)

11111111 11111111 11111111 11------ 255.255.255.192 (subnet mask)

===================================== logical\_AND 00001001 01000011 00100110 00------ = 9.67.38.0 (subnet base address)

This leaves a host address of:

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

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

The subnet number is:

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

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

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

##### Variable length subnetting example

Consider a corporation that has been assigned the Class C network

165.214.32.0. The corporation has the requirement to split this address range into five separate networks each with the following number of hosts:

Subnet 1: 50 hosts

Subnet 2: 50 hosts

Subnet 3: 50 hosts

Subnet 4: 30 hosts

Subnet 5: 30 hosts

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

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

##### Determining the subnet mask

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

##### Addressing routers and multihomed hosts

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

#### 3.1.3 IP routing

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

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

Hosts that are directly attached to one of the physical networks to which the router is attached.

Hosts or networks for which the router has been given explicit definitions.

Hosts or networks for which the router has received an ICMP redirect message.

A default for all other destinations.

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

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

##### Direct routing

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

##### Indirect routing

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

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

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

Host D

Host C

Host B

Host A

*Figure 3-5 IP: Direct and indirect routes*

##### IP routing table

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

Destination IP network addresses

Routes to next gateways

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

The direct routes describing locally attached networks

The indirect routes describing networks reachable through one or more gateways

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

Figure3-6 presents a sample network.

*Figure 3-6 IP: Routing table scenario*

Host D

Host C

Host E

Host F

Host B

Host A

128.15

129.7

128.10

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

*Table 3-2 Host D sample entries*

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

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

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

*Table 3-3 Host F sample entries*

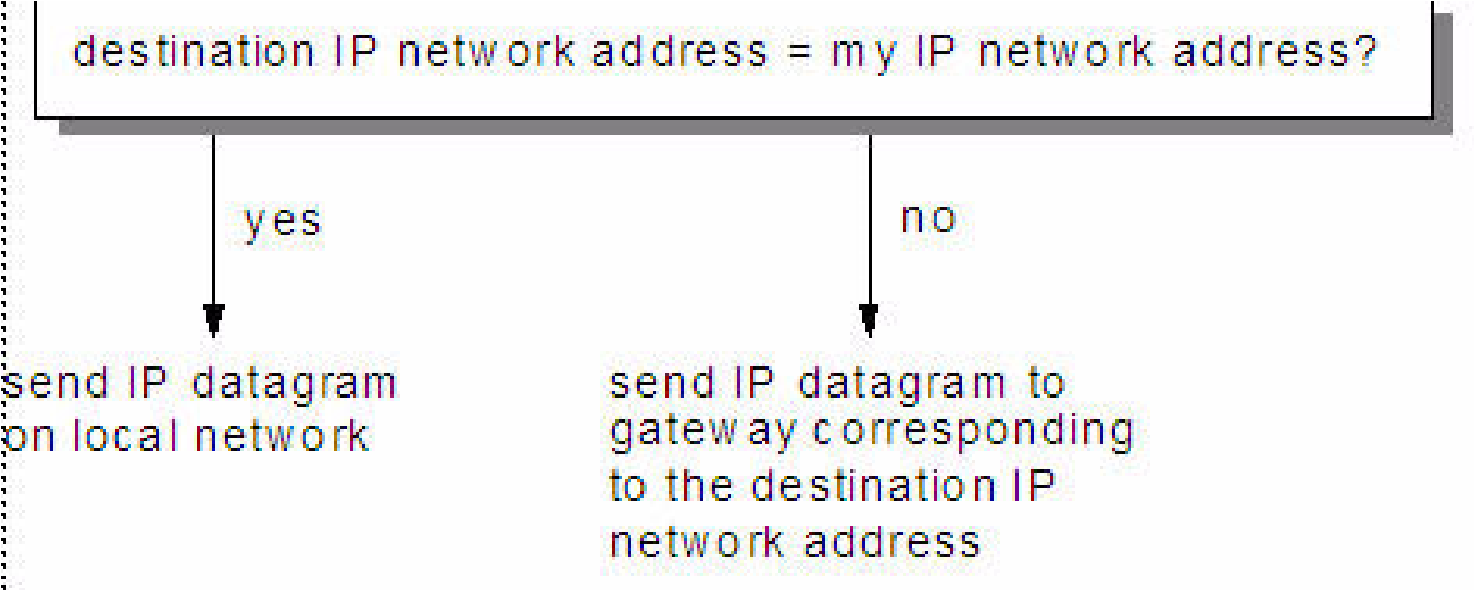
|  |  |  |
| --- | --- | --- |
| **Destination** | **Router** | **Interface** |
| 129.7.0.0 | F | wan0 |
| default | E | wan0 |
| 127.0.0.1 | loopback | lo |

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

##### IP routing algorithm

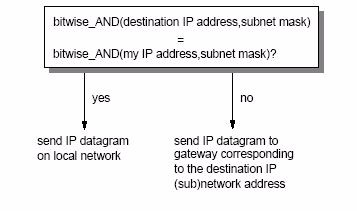
IP uses a unique algorithm to route datagrams, as illustrated in Figure 3-7.

*Figure 3-7 IP: Routing without subnets*



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

*Figure 3-8 IP: Routing with subnets*



Some implications of this change include:

This algorithm represents a change to the general IP algorithm. Therefore, to be able to operate this way, the particular gateway must contain the new algorithm. Some implementations might still use the general algorithm, and will not function within a subnetted network, although they can still communicate with hosts in other networks that are subnetted.

As IP routing is used in all of the hosts (and not just the routers), all of the hosts in the subnet must have:

* An IP routing algorithm that supports subnetting
* The same subnet mask (unless subnets are formed within the subnet)

If the IP implementation on any of the hosts does not support subnetting, that host will be able to communicate with any host in its own subnet but not with any machine on another subnet within the same network. This is because the host sees only one IP network and its routing cannot differentiate between an IP datagram directed to a host on the local subnet and a datagram that should be sent through a router to a different subnet.

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

Figure 3-9 illustrates the entire IP routing algorithm.

Bitwise AND local interface(s)

with local\_subnet\_mask(s)

Yes

Yes

Yes

Take destination IP

address

No

No

No

Send ICMP error message

"network unreachable"

Bitwise AND dest\_IP\_addr

with local\_subnet\_mask(s)

Is there a match?

Is there an indirect route

entry?

Is a default route

specified?

Deliver indirectly to the

default router's IP address

Deliver directly using the

corresponding local

interface

Deliver indirectly to the

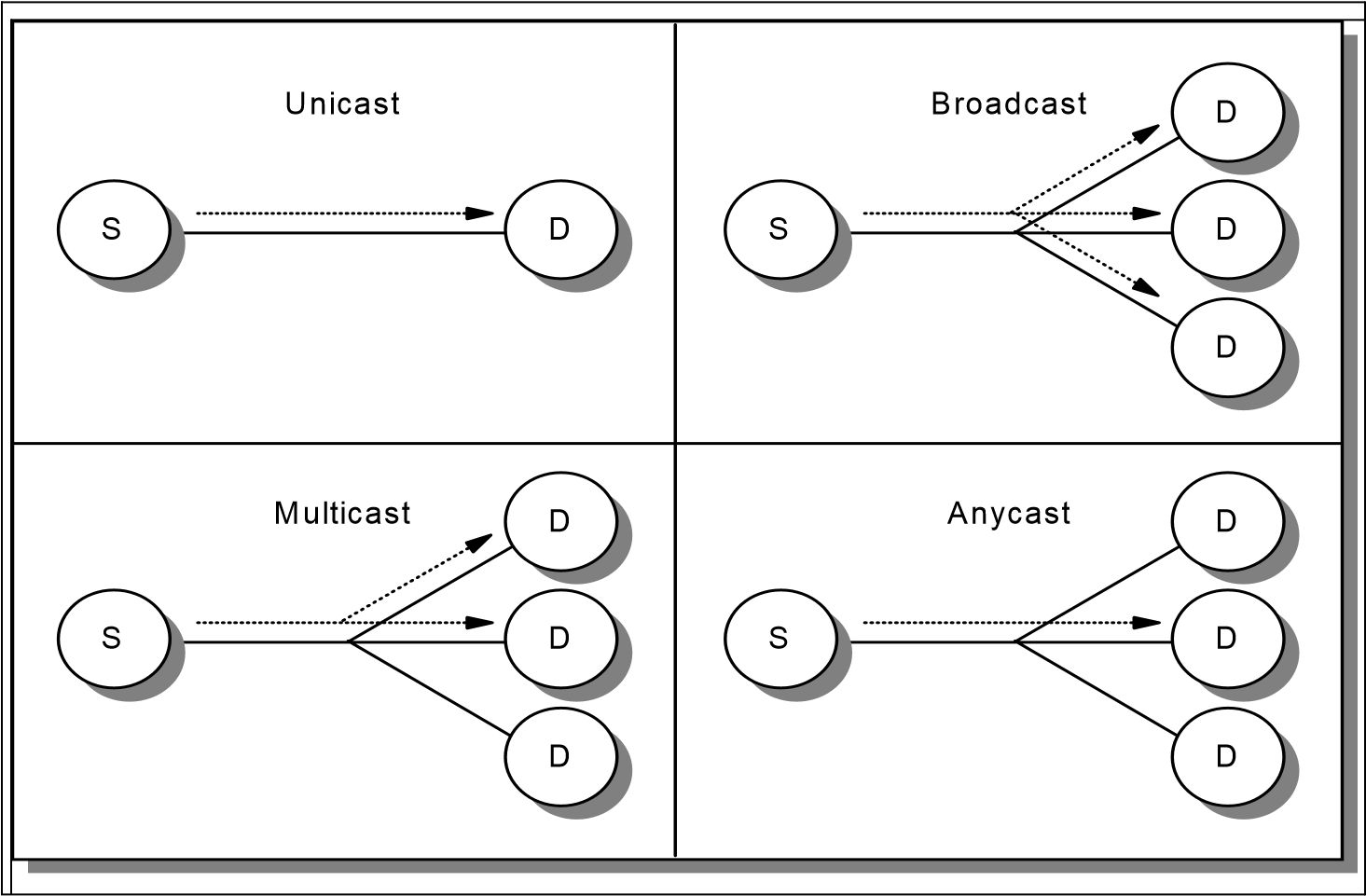
corresponding router's IP

address

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

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

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



*Figure 3-10 IP: Packet delivery modes*

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

##### Broadcasting

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

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

information. Routers do not forward this packet.

One exception to this rule is called *BOOTP forwarding*. The BOOTP protocol uses the limited broadcast address to allow a diskless workstation to contact a boot server. BOOTP forwarding is a configuration option available on some routers. Without this facility, a separate BOOTP server is required on each subnet (refer to 3.6, “Bootstrap Protocol (BOOTP)” on page 125).

Network-directed broadcast address: This is used in an unsubnetted environment. The network number is a valid network number and the host number is all ones (for example, 128.2.255.255). This address refers to all hosts on the specified network. Routers should forward these broadcast messages. This is used in ARP requests (refer to 3.4, “Address Resolution Protocol (ARP)” on page 119) on unsubnetted networks.

Subnet-directed broadcast address: If the network number is a valid network number, the subnet number is a valid subnet number, and the host number is all ones, the address refers to all hosts on the specified subnet. Because the sender's subnet and the target subnet might have a different subnet mask, the sender must somehow determine the subnet mask in use at the target. The broadcast is performed by the router that receives the datagram into the subnet.

All-subnets-directed broadcast address: If the network number is a valid network number, the network is subnetted, and the local part is all ones (for example, 128.2.255.255), the address refers to all hosts on all subnets in the specified network. In principle, routers can propagate broadcasts for all subnets but are not required to do so. In practice, they do not. There are very few circumstances where such a broadcast is desirable. If misconfigured, it can lead to problems. Consider the misconfigured host 9.180.214.114 in a subnetted Class A network. If the device was configured with the address 9.255.255.255 as a local broadcast address instead of 9.180.214.255, all of the routers in the network will forward the request to all clients.

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

##### Multicasting

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

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

##### Anycasting

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

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

#### 3.1.5 The IP address exhaustion problem

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

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

In May 1996, all Class A addresses were either allocated or assigned, as well as 61.95 percent of Class B and 36.44 percent of Class C IP network addresses.

The terms assigned and allocated in this context have the following meanings:

Assigned: The number of network numbers in use. The Class C figures are somewhat inaccurate, because the figures do not include many Class C

networks in Europe, which were allocated to RIPE and subsequently assigned but which are still recorded as allocated.

Allocated: This includes all of the assigned networks and additionally, those networks that have either been reserved by IANA (for example, the 63 Class A networks are all reserved by IANA) or have been allocated to regional registries by IANA and will subsequently be assigned by those registries.

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

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

The upper half of the Class A address space (network numbers 64 to 127) is reserved indefinitely to allow for the possibility of using it for transition to a new numbering scheme.

Class B networks are only assigned to organizations that can clearly demonstrate a need for them. The same is, of course, true for Class A networks. The requirements for Class B networks are that the requesting organization:

* Has a subnetting plan that documents more than 32 subnets within its organizational network
* Has more than 4096 hosts

Any requirements for a Class A network are handled on an individual case basis.

Organizations that do not fulfill the requirements for a Class B network are assigned a consecutively numbered block of Class C network numbers.

The lower half of the Class C address space (network numbers 192.0.0 through 207.255.255) is divided into eight blocks, which are allocated to regional authorities as follows:

|  |  |  |
| --- | --- | --- |
| **192.0.0 - 193.255.255** | Multi-regional | |
| **194.0.0 - 195.255.255** | Europe | |
| **196.0.0 - 197.255.255** | Others | |
| **198.0.0 - 199.255.255** | North America | |
| **200.0.0 - 201.255.255** | Central and South America | |
| **202.0.0 - 203.255.255** | Pacific Rim | |
| **204.0.0 - 205.255.255 206.0.0 - 207.255.255 208.0.0 - 209.255.255**  **210.0.0 - 211.255.255** | Others  Others  ARIN1  APNIC | |
| **212.0.0 - 213.255.255** | RIPE NCC |
| **214.0.0 - 215.255.255** | US Department of Defense |
| **216.0.0 - 216.255.255** | ARIN |
| **217.0.0 - 217.255.255** | RIPE NCC |
| **218.0.0 - 218.255.255** | APNIC |
| **219.0.0 - 222.255.255** | APNIC |

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

Where an organization has a range of Class C network numbers, the range provided is assigned as a *bit-wise contiguous* range of network numbers, and the number of networks in the range is a power of 2. That is, all IP addresses in the range have a common prefix, and every address with that prefix is within the range. For example, a European organization requiring 1500 IP addresses would be assigned eight Class C network numbers (2048 IP addresses) from the number space reserved for European networks (194.0.0 through 195.255.255) and the first of these network numbers would be divisible by eight. A range of addresses satisfying these rules is 194.32.136 through 194.32.143, in which case the range consists of all of the IP addresses with the 21-bit prefix 194.32.136, or B '110000100010000010001'.

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

of small networks, and the organization should request 10 different Class C numbers, one for each LAN.

1 Information for this and the following numbers in this list are from:

<http://www.iana.org/assignments/ipv4-address-space>

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

#### 3.1.6 Intranets: Private IP addresses

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

10.0.0.0: A single Class A network

172.16.0.0 through 172.31.0.0: 16 contiguous Class B networks

192.168.0.0 through 192.168.255.0: 256 contiguous Class C networks

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

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

#### 3.1.7 Network Address Translation (NAT)

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

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

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

##### Traditional NAT

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

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

##### Basic NAT

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

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



Non-Secure

a.b.1.0/24

a.b.1.1

10.0.1.1

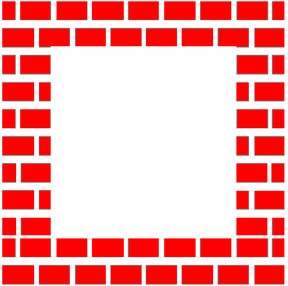
Secure

src=a.b.1.1 dest=a.b.2.1

src=a.b.1.1 dest=10.0.1.1



10.0.0.0/8



NAT

Filtering

TCP/UDP

IP/ICMP



NAT Configuration

**RESERVE a.b.2.0 255.255.255.0**

**TRANSLATE 10.0.0.0 255.0.0.0**



Filtering Rules

Based on non-translated

IP addresses (10.x.x.x)

*Figure 3-11 Basic Network Address Translation (NAT)*

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



Non-Secure

a.b.1.0/24

a.b.1.1

Looks like a

normal router

src=a.b.1.1 dest=a.b.2.1

a.b.2.0/24

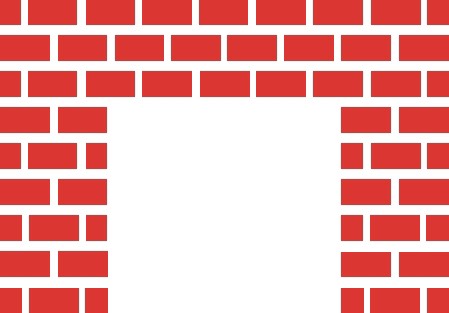
a.b.2.1

Secure

*Figure 3-12 NAT seen from the external network*

##### Basic NAT translation mechanism

For each outgoing IP packet, the source address is checked by the NAT configuration rules. If a rule matches the source address, the address is translated to a global address from the address pool. The predefined address pool contains the addresses that NAT can use for translation. For each incoming packet, the destination address is checked if it is used by NAT. When this is true, the address is translated to the original internal address. Figure 3-13 shows the Basic NAT configuration.



To be translated

Exclude

Secure network

Non-secure network

Reserve

N

A

T

Exclude

Firewall

Map

Pool

*Figure 3-13 Basic NAT configuration*

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

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

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

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

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

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

##### Network Address Port Translation (NAPT)

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

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Transition Table  10.10.10.11:80 = a.b.65.1:8000  10.10.10.12:80 = a.b.65.1:8001 |  |  |
| External  a.b.65.0 /30  a.b.65.3 /30 |  | Internal  10.10.10.0 /24 | 10.10.10.11 /24  10.10.10.12 /24 |
| NAPT |

*Figure 3-14 Network Address Port Translation*

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

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

##### NAT limitations

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

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

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

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

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

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

#### 3.1.8 Classless Inter-Domain Routing (CIDR)

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

router. The same environment, if addressed as a range of Class C networks, requires 16 entries.

The solution to this problem is called Classless Inter-Domain Routing (CIDR).

CIDR is described in RFCs 1518 to 1520. CIDR does not route according to the

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

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

11000000 00100000 10001000 00000000 = 192.32.136.0 (Class C address)

11111111 11111111 11111--- -------- 255.255.248.0 (network mask)

===================================== logical\_AND

11000000 00100000 10001--- -------- = 192.32.136 (IP prefix)

11000000 00100000 10001111 00000000 = 192.32.143.0 (Class C address)

11111111 11111111 11111--- -------- 255.255.248.0 (network mask)

===================================== logical\_AND

11000000 00100000 10001--- -------- = 192.32.136 (same IP prefix)

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

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

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

IP address assignment reflects the physical topology of the network and not the organizational topology. Wherever organizational and administrative boundaries do not match the network topology, they should *not* be used for the assignment of IP addresses.

In general, network topology will closely follow continental and national boundaries. Therefore, IP addresses should be assigned on this basis.

There will be a relatively small set of networks that carry a large amount of traffic between routing domains. These networks will be interconnected in a non-hierarchical way that crosses national boundaries. These networks are referred to as *transit routing domains (TRDs)* Each TRD will have a unique IP prefix. TRDs will not be organized in a hierarchical way when there is no appropriate hierarchy. However, whenever a TRD is wholly within a continental boundary, its IP prefix should be an extension of the continental IP prefix.

There will be many organizations that have attachments to other organizations that are for the private use of those two organizations. The attachments do not carry traffic intended for other domains (transit traffic). Such private connections do not have a significant effect on the routing topology and can be ignored.

The great majority of routing domains will be single-homed. That is, they will be attached to a single TRD. They should be assigned addresses that begin with that TRD's IP prefix. All of the addresses for all single-homed domains attached to a TRD can therefore be aggregated into a single routing table entry for all domains outside that TRD.

There are a number of address assignment schemes that can be used for multihomed domains. These include:

* The use of a single IP prefix for the domain. External routers must have an entry for the organization that lies partly or wholly outside the normal hierarchy. Where a domain is multihomed, but all of the attached TRDs themselves are topologically nearby, it is appropriate for the domain's IP prefix to include those bits common to all of the attached TRDs. For example, if all of the TRDs were wholly within the United States, an IP prefix implying an exclusively North American domain is appropriate.
* The use of one IP prefix for each attached TRD with hosts in the domain having IP addresses containing the IP prefix of the most appropriate TRD. The organization appears to be a set of routing domains.
* Assigning an IP prefix from one of the attached TRDs. This TRD becomes a default TRD for the domain but other domains can explicitly route by one of the alternative TRDs.
* The use of IP prefixes to refer to sets of multihomed domains having the TRD attachments. For example, there can be an IP prefix to refer to single-homed domains attached to network A, one to refer to

single-homed domains attached to network B, and one to refer to dual-homed domains attached to networks A and B.

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

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

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

##### CIDR implementation

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

Type 1: Those providers that cannot employ any default inter-domain routing.

Type 2: Those providers that use default inter-domain routing but require explicit routes for a substantial proportion of the assigned IP network numbers.

Type 3: Those providers that use default inter-domain routing and supplement it with a small number of explicit routes.

Type 4: Those providers that perform inter-domain routing using only default routes.

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

#### 3.1.9 IP datagram

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | header | data |   base IP datagram...   |  |  | | --- | --- | | physical network header | IP datagram as data |   encapsulated within the physical network's frame |

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

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

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

##### IP datagram format

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 1 2 3  0 4 8 6 9 4 1   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | VERS |  | HLEN | | Service Type | Total | | Length | |  |  | ID | | | FLG | | Fragment Offset | |  | TTL |  | Protocol | |  | | Header Checksum | |  |  | Source IP Ad | | | | | dress | |  |  | Destination IP | | | | | Address | |  |  | IP Options | | | |  | Padding | |  |  | Data …  …  … | | | | |  | |

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

Where:

VERS: The field contains the IP protocol version. The current version is 4. Version 5 is an experimental version. Version 6 is the version for IPv6 (see 9.2, “The IPv6 header format” on page 330).

HLEN: The length of the IP header counted in 32-bit quantities. This does not include the data field.

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

|  |  |  |  |
| --- | --- | --- | --- |
| 0 1 2 3 4 5 6 7   |  |  |  | | --- | --- | --- | | precedence | TOS | MBZ | |

*Figure 3-18 IP: Service type*

Where:

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

A detailed description of the type of service is in the RFC 1349 (refer to 8.1, “Why QoS?” on page 288).

* MBZ:Reserved for future use.

Total Length: The total length of the datagram, header and data.

Identification: A unique number assigned by the sender to aid in reassembling a fragmented datagram. Each fragment of a datagram has the same identification number.

Flags: This field contains control flags illustrated in Figure 3-19.

0

0

1

2

D

F

M

F

*Figure 3-19 IP: Flags*

Where:

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

Fragment Offset: This is used to aid the reassembly of the full datagram. The value in this field contains the number of 64-bit segments (header bytes are not counted) contained in earlier fragments. If this is the first (or only) fragment, this field contains a value of zero.

Time to Live: This field specifies the time (in seconds) the datagram is allowed to travel. Theoretically, each router processing this datagram is supposed to subtract its processing time from this field. In practise, a router processes the datagram in less than 1 second. Therefore, the router subtracts one from the value in this field. The TTL becomes a hop-count metric rather than a time metric. When the value reaches zero, it is assumed that this datagram has been traveling in a closed loop and is discarded. The initial value should be set by the higher-level protocol that creates the datagram.

Protocol Number: This field indicates the higher-level protocol to which IP should deliver the data in this datagram. These include:

* 0: Reserved
* 1: Internet Control Message Protocol (ICMP)
* 2: Internet Group Management Protocol (IGMP)
* 3: Gateway-to-Gateway Protocol (GGP)
* 4: IP (IP encapsulation)
* 5: Stream
* 6: Transmission Control Protocol (TCP)
* 8: Exterior Gateway Protocol (EGP)
* 9: Interior Gateway Protocol (IGP)
* 17: User Datagram Protocol (UDP)
* 41:Simple Internet Protocol (SIP)
* 50: SIPP Encap Security Payload (ESP)
* 51: SIPP Authentication Header (AH)
* 89: Open Shortest Path First (OSPF) IGP

The complete list is in STD 2 – Assigned Internet Numbers.

Header Checksum: This field is a checksum for the information contained in the header. If the header checksum does not match the contents, the datagram is discarded.

Source IP Address: The 32-bit IP address of the host sending this datagram. Destination IP Address: The 32-bit IP address of the destination host for this datagram.

Options: An IP implementation is not required to be capable of generating options in a datagram. However, all IP implementations are required to be able to process datagrams containing options. The Options field is variable in length (there can be zero or more options). There are two option formats. The format for each is dependent on the value of the option number found in the first octet:

* A type octet alone is illustrated in Figure 3-20.

|  |  |
| --- | --- |
| |  | | --- | | type |   1 byte |

*Figure 3-20 IP: A type byte*

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

type

length

option data...

/ /

/ /

1

byte

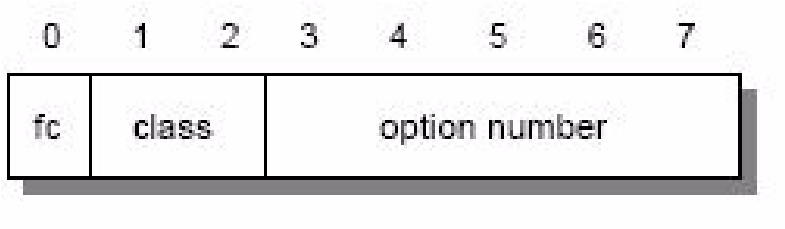
length - 2 bytes

1

byte

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

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



*Figure 3-22 IP: The type byte structure*

Where:

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

Padding: If an option is used, the datagram is padded with all-zero octets up to the next 32-bit boundary.

Data: The data contained in the datagram. It is passed to the higher-level protocol specified in the protocol field.

##### Fragmentation

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

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

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

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

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

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

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

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

##### IP datagram routing options

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

###### Loose source routing

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

10000011

length

pointer

route data

//

/

/

*Figure 3-23 IP: Loose source routing option*

The fields of this header include:

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

**Length** This field contains the length of this option field,

including the type and length fields.

|  |  |
| --- | --- |
| **Pointer** | This field points to the option data at the next IP address to be processed. It is counted relative to the beginning of the option, so its minimum value is four. If the pointer is greater than the length of the option, the end of the source route is reached and further routing is to be based on the destination IP address (as for datagrams without this option). |
| **Route data** | This field contains a series of 32-bit IP addresses. |

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

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

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

###### Strict source routing

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

*Figure 3-24 IP: Strict source routing option*

100001001

length

pointer

route data

//

/

/

Where:

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

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

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

**Route data** A series of 32-bit IP addresses.

###### Record route

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

00000111

length

pointer

route data

//

/

/

*Figure 3-25 IP: Record route option*

Where:

|  |  |
| --- | --- |
| **0000111 (Decimal 7)** | The value of the option type byte for record route |
| **Length** | This information is described in “Loose source routing” on page 105. |
| **Pointer** | This information is described in “Loose source routing” on page 105. |
| **Route data** | A series of 32-bit IP addresses. |

##### Internet time stamp

A time stamp is an option forcing some (or all) of the routers along the route to the destination to put a time stamp in the option data. The time stamps are

measured in seconds and can be used for debugging purposes. They cannot be used for performance measurement for two reasons:

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

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 24 28   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 01000100 | length |  | pointer | oflw | flag | |  | IP address | |  |  | | |  | time st | amp |  |  | | |  | ... |  |  |  | | |

*Figure 3-26 IP: Internet time stamp option*

Where:

|  |  |  |
| --- | --- | --- |
| **01000100 (Decimal 68)** | | This field is the value of the option type for the internet time stamp option. |
| **Length** |  | This field contains the total length of this option, including the type and length fields. |
| **Pointer** |  | This field points to the next time stamp to be processed (first free time stamp). |
| **Oflw (overflow)** |  | This field contains the number of devices that cannot register time stamps due to a lack of space in the data field. |
| **Flag** |  | This field is a 4-bit value that indicates how time stamps are to be registered:   1. Time stamps only, stored in consecutive 32-bit words. 2. Each time stamp is preceded by the IP address of the registering device. 3. The IP address fields are prespecified; an IP device only registers when it finds its own address in the list. |
| **Time stamp** | A 32-bit time stamp recorded in milliseconds since midnight UT (GMT). | |

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

### 3.2 Internet Control Message Protocol (ICMP)

ICMP is a standard protocol with STD number 5. That standard also includes IP

(see 3.1, “Internet Protocol (IP)” on page 68) and IGMP (see 6.2, “Internet Group

Management Protocol (IGMP)” on page 241). Its status is required. It is described in RFC 792 with updates in RFC 950. ICMPv6 used for IPv6 is discussed in 9.3, “Internet Control Message Protocol Version 6 (ICMPv6)” on page 352.

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

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

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

ICMP uses IP as though ICMP were a higher-level protocol (that is, ICMP messages are encapsulated in IP datagrams). However, ICMP is an integral part of IP and must be implemented by every IP module.

ICMP is used to report errors, *not* to make IP reliable. Datagrams can still be undelivered without any report on their loss. Reliability must be implemented by the higher-level protocols using IP services.

ICMP cannot be used to report errors with ICMP messages. This avoids infinite repetitions. ICMP responses are sent in response to ICMP query messages (ICMP types 0, 8, 9, 10, and 13 through 18).

For fragmented datagrams, ICMP messages are only sent about errors with the first fragment. That is, ICMP messages never refer to an IP datagram with a non-zero fragment offset field.

ICMP messages are never sent in response to datagrams with a broadcast or a multicast destination address.

ICMP messages are never sent in response to a datagram that does not have a source IP address representing a unique host. That is, the source address cannot be zero, a loopback address, a broadcast address, or a multicast address.

RFC 792 states that ICMP messages *can* be generated to report IP datagram processing errors. However, this is not required. In practice, routers will almost always generate ICMP messages for errors. For destination hosts, ICMP message generation is implementation dependent.

#### 3.2.1 ICMP messages

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

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  |  |  | | --- | --- | --- | --- | | identifier | sequence number |  | checksum | | ICMP data (depending on the type of mes | | sage) |  | |

*Figure 3-27 ICMP: Message format*

The message contains the following components:

**Type** Specifies the type of the message:

**0** Echo reply

1. Destination unreachable
2. Source quench
3. Redirect
4. Echo
5. Router advertisement
6. Router solicitation
7. Time exceeded
8. Parameter problem
9. Time stamp request
10. Time stamp reply
11. Address mask request
12. Address mask reply

**30** Traceroute

1. Domain name request)
2. Domain name reply)

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

|  |  |
| --- | --- |
| **Code** | Contains the error code for the datagram reported by this  ICMP message. The interpretation is dependent on the |
| **Checksum** | message type.    Contains the checksum for the ICMP message starting with the ICMP Type field. If the checksum does not match the contents, the datagram is discarded. |

**Data** Contains information for this ICMP message. Typically, it

will contain the portion of the original IP message for which this ICMP message was generated.

Each of the ICMP messages is described individually.

##### Echo (8) and Echo Reply (0)

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identifier | sequence number | |  | data ... | |

*Figure 3-28 Echo and Echo Reply*

##### Destination Unreachable (3)

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

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

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | unused (zero) | | IP header - 64 bits of original data of the datagram | |

*Figure 3-29 ICMP: Destination Unreachable*

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

1. Network unreachable
2. Host unreachable
3. Protocol unreachable
4. Port unreachable
5. Fragmentation needed but the Do Not Fragment bit was set
6. Source route failed
7. Destination network unknown
8. Destination host unknown
9. Source host isolated (obsolete)
10. Destination network administratively prohibited
11. Destination host administratively prohibited
12. Network unreachable for this type of service
13. Host unreachable for this type of service
14. Communication administratively prohibited by filtering
15. Host precedence violation
16. Precedence cutoff in effect

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | unused (zero) | link MTU | | IP header - 64 bits of original data of the datagram | | |

*Figure 3-30 ICMP: Fragmentation required with link MTU*

##### Source Quench (4)

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

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

The ICMP header code field is always zero.

See Figure 3-31 for more details.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | unused (zero) | | IP header - 64 bits of original data of the datagram | |

*Figure 3-31 ICMP: Source Quench*

##### Redirect (5)

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

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

1. Network redirect
2. Host redirect
3. Network redirect for this type of service **3** Host redirect for this type of service See Figure 3-32 for more details.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | router IP address | | IP header - 64 bits of original data of the datagram | |

*Figure 3-32 ICMP: Redirect*

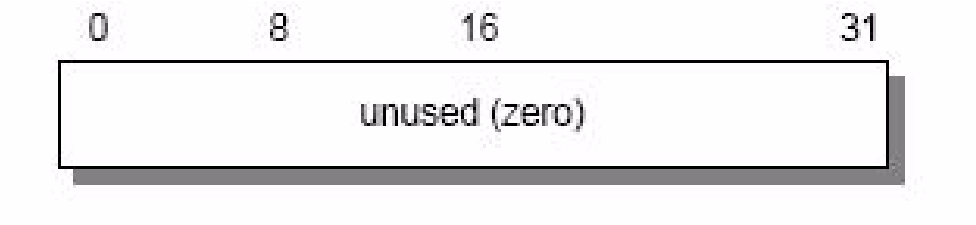
##### Router Advertisement (9) and Router Solicitation (10)

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31 | | | | | | | |
|  |  | |  |  | |  | |
|  | number | entry length | TTL | |  |  |
|  |  | | | |
| router address 1 | | | |  |
|  |  | | | |
| preference level 1 | | | |  |
|  |  | | | |
|  | | | |  |
| / | / |  | | | / | / |
|  |  |  |  |
|  | | | | |
|  | router address n | | | |  |
|  | preference level n | | | |
|  |  | | | |
|  | | | | |

*Figure 3-33 ICMP: Router Advertisement*

*Figure 3-34 ICMP: Router Solicitation*



Where:

|  |  |
| --- | --- |
| **Number** | The number of entries in the message. |
| **Entry length** | The length of an entry in 32-bit units. This is 2 (32 bits for the IP address and 32 bits for the preference value). |
| **TTL** | The number of seconds that an entry will be considered valid. |
| **Router address** | One of the sender's IP addresses. |
| **Preference level** | A signed 32-bit level indicating the preference to be assigned to this address when selecting a default router. Each router on a subnet is responsible for advertising its own preference level. Larger values imply higher preference; smaller values imply lower. The default is zero, which is in the middle of the possible range. A value of X'80000000' (-231) indicates the router should never be used as a default router. |

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

These two messages are used if a host or a router supports the router discovery protocol. Routers periodically advertise their IP addresses on those subnets where they are configured to do so. Advertisements are made on the all-systems multicast address (224.0.0.1) or the limited broadcast address

(255.255.255.255). The default behavior is to send advertisements every 10 minutes with a TTL value of 1800 (30 minutes). Routers also reply to solicitation messages they receive. They might reply directly to the soliciting host, or they might wait a short random interval and reply with a multicast.

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

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

##### Time Exceeded (11)

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

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

1. Transit TTL exceeded
2. Reassembly TTL exceeded

See Figure 3-35 for more details.

|  |  |  |
| --- | --- | --- |
| 0 8 16 31   |  | | --- | | unused (zero) | | IP header - 64 bits of original data of the datagram | |

*Figure 3-35 ICMP: Time Exceeded*

##### Parameter Problem (12)

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

1. Unspecified error
2. Required option missing

See Figure 3-36 for more details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | pointer | unused (zero) | | IP header - 64 bits of original data of the datagram | | |

*Figure 3-36 ICMP: Parameter Problem*

##### Timestamp Request (13) and Timestamp Reply (14)

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

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

See Figure 3-37 for details.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identifier | sequence number | | originate timestamp | | | receive timestamp | | | transmit timestamp | | |

*Figure 3-37 ICMP: Timestamp Request and Timestamp Reply*

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

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 8 16 31   |  |  | | --- | --- | | identifier | sequence number | | subnet address mask | | |

*Figure 3-38 ICMP: Address Mask Request and Reply*

#### 3.2.2 ICMP applications

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

##### Ping

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

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

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

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

Most platforms allow you to specify the following values:

|  |  |
| --- | --- |
| **Size** | The size of the data portion of the packet. |
| **Packets** | The number of packets to send. |
| **Count** | The number of echo requests to send. |
| **Record routes** | Record the route per count hop. | |
| **Time stamp** | Time stamp each count hop. | |
| **Endless ping** | Ping until manually stopped. | |
| **Resolve address** | Resolve the host address to the host name. | |
| **Time to Live (TTL)** | The time (in seconds) the datagram is allowed to travel. | |

**Type of Service (TOS)** The type of internet service quality.

**Host-list** Loose source route or strict source route of host lists.

**Timeout** The timeout to wait for each reply. **No fragmentation** The fragment flag is not set.

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

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

ping loopback: Verifies the operation of the base TCP/IP software.

ping my-IP-address: Verifies whether the physical network device can be addressed.

ping a-remote-IP-address: Verifies whether the network can be accessed.

ping a-remote-host-name: Verifies the operation of the name server (or the flat namespace resolver, depending on the installation).

##### Traceroute

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

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

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

### 3.3 Internet Group Management Protocol (IGMP)

IGMP is a standard protocol with STD number 5. That standard also includes IP (see 3.1, “Internet Protocol (IP)” on page 68) and ICMP (see 3.2, “Internet

Control Message Protocol (ICMP)” on page 109). Its status is recommended. It is described in RFC 1112 with updates in RFC 2236.

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

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

### 3.4 Address Resolution Protocol (ARP)

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

RFC 826.

#### 3.4.1 ARP overview

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

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

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

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

ARP was designed to be used on networks that support hardware broadcast.

This means, for example, that ARP will not work on an X.25 network.

#### 3.4.2 ARP detailed concept

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

##### ARP packet generation

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  |  | physical layer header | | | A  R  P  P a c k e t | hardware address space | | | | Protocol address space | | | | hardware address byte length (n) | | Protocol address byte length (m) | | operation code | | | | hardware address of sender | | | | protocol address of sender | | | | hardware address of target | | | | protocol address of target | | |   x bytes  2 bytes  2 bytes 2bytes  2 bytes n bytes m bytes n bytes  m bytes |

*Figure 3-39 ARP: Request/reply packet*

Where:

Hardware address space: Specifies the type of hardware; examples are Ethernet or Packet Radio Net.

Protocol address space: Specifies the type of protocol, same as the EtherType field in the IEEE 802 header (IP or ARP).

Hardware address length: Specifies the length (in bytes) of the hardware addresses in this packet. For IEEE 802.3 and IEEE 802.5, this is 6.

Protocol address length: Specifies the length (in bytes) of the protocol addresses in this packet. For IP, this is 4.

Operation code: Specifies whether this is an ARP request (1) or reply (2).

Source/target hardware address: Contains the physical network hardware addresses. For IEEE 802.3, these are 48-bit addresses.

Source/target protocol address: Contains the protocol addresses. For TCP/IP, these are the 32-bit IP addresses.

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

##### ARP packet reception

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

Yes

Yes

Yes

Yes

Yes

Yes

No

(

discard

)

Do I have the specified

hardware type?

No

(

discard

)

No

No

)

discard

(

No

No

(

discard

)

End

Do I speak the specified

protocol?

Is the pair <protocol type,

sender protocol address>

already in my table?

Am I the target protocol

address?

Is flag = false?

Is the opcode a request?

Add the triplet <protocol

type, sender protocol and

sender hardware> to

table.

Set flag = false.

Update the table with the

sender hardware address.

Set flag=true.

Swap source and target

addressesin the ARP

packet. Put my local

addresses in the source

address fields. Send back

ARP packet as an ARP

reply to the requesting

host.

*Figure 3-40 ARP: Packet reception*

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

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

#### 3.4.3 ARP and subnets

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

#### 3.4.4 Proxy-ARP or transparent subnetting

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

##### Proxy-ARP concept

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

A

R

B

*Figure 3-41 ARP: Hosts interconnected by a router*

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

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

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

The result is transparent subnetting:

Normal hosts (such as A and B) do not know about subnetting, so they use the “old” IP routing algorithm.

The routers between subnets have to:

* Use the subnet IP routing algorithm.
* Use a modified ARP module, which can reply on behalf of other hosts.

See Figure 3-42 for more details.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | |  | |
|  | A |  |
|  |
| "o ld " IP ro u tin g |  | B |
|  |
|  |  |  |  |
|  | R |  |
| IP s u b n e t ro u tin g a n d m o d ifie d A R P |

*Figure 3-42 ARP: Proxy-ARP router*

### 3.5 Reverse Address Resolution Protocol (RARP)

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

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

#### 3.5.1 RARP concept

The reverse address resolution is performed the same way as the ARP address

resolution. The same packet format (see Figure 3-39 on page 121) is used as for

ARP.

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

1. For the RARP request
2. For the RARP reply

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

Some differences arise from the concept of RARP itself:

ARP only assumes that every host knows the mapping between its own hardware address and protocol address. RARP requires one or more server hosts in the network to maintain a database of mappings between hardware addresses and protocol addresses so that they will be able to reply to requests from client hosts.

Due to the size this database can take, part of the server function is usually implemented outside the adapter's microcode, with optionally a small cache in the microcode. The microcode part is then only responsible for reception and transmission of the RARP frames, the RARP mapping itself being taken care of by server software running as a normal process on the host machine.

The nature of this database also requires some software to create and update the database manually.

If there are multiple RARP servers in the network, the RARP requester only uses the first RARP reply received on its broadcast RARP request and discards the others.

### 3.6 Bootstrap Protocol (BOOTP)

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

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

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

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

The BOOTP process involves the following steps:

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

hardware address matching the IP address. ARP will be used to determine it as normal.

* + If the client does not know its own IP address (it was 0.0.0.0 in the BOOTP request), the server must concern itself with its own ARP cache.
  + ARP on the server cannot be used to find the hardware address of the client because the client does not know its IP address and so cannot reply to an ARP request. This is called the “chicken and egg” problem. There are two possible solutions:
    - If the server has a mechanism for directly updating its own ARP cache without using ARP itself, it does so and then sends the datagram directly.
    - If the server cannot update its own ARP cache, it must send a broadcast reply.

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 8 16 24 31   |  |  |  |  |  | | --- | --- | --- | --- | --- | | code |  | H/W type | length | hops | |  |  | transaction ID | |  | | seconds | |  | flags | field | |  | | client IP address | |  | |  | | your IP address | |  | |  | | server IP address | |  | |  | | router IP address | |  | |  | | client hardware address (16 bytes) | |  | |  | | server host name (64 bytes) | |  | |  | | boot file name (128 bytes) | |  | |  | | vendor-specific area (64 bytes) | |  | |

*Figure 3-43 BOOTP message format* Where:

|  |  |
| --- | --- |
| **Code** | Indicates a request or a reply:   1. Request 2. Reply |
| **H/W type** | The type of hardware, for example: |

**1** Ethernet

**6** IEEE 802 Networks

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

|  |  |
| --- | --- |
| **Length** | Hardware address length in bytes. Ethernet and token ring both use 6, for example. |
| **Hops** | The client sets this to 0.  It is incremented by a router that relays the request to another server and is used to identify loops. RFC 951 suggests that a value of 3 indicates a loop. |
| **Transaction ID** | A random number used to match this boot request with the response it generates. |
| **Seconds** | Set by the client. It is the elapsed time in seconds since the client started its boot process. |
| **Flags field** | The most significant bit of the flags field is used as a broadcast flag. All other bits must be set to zero; they are reserved for future use. Normally, BOOTP servers attempt to deliver BOOTREPLY messages directly to a client using unicast delivery. The destination address in the IP header is set to the BOOTP *your IP address* and the MAC address is set to the BOOTP *client hardware address*. If a host is unable to receive a unicast IP datagram until it knows its IP address, this broadcast bit must be set to indicate to the server that the BOOTREPLY must be sent as an IP and MAC broadcast. Otherwise, this bit must be set to zero. |
| **Client IP address** | Set by the client, either to its known IP address or 0.0.0.0. |
| **Your IP address** | Set by the server if the client IP address field was 0.0.0.0. |
| **Server IP address** | Set by the server. |
| **Router IP address** | This is the address of a BOOTP relay agent, *not* a general IP router to be used by the client. It is set by the forwarding agent when BOOTP forwarding is used (see 3.6.1, “BOOTP forwarding” on page 129). |

#### Client hardware address

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

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

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

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

#### 3.6.1 BOOTP forwarding

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

A router will normally discard any datagrams containing illegal source addresses, such as 0.0.0.0, which is used by a BOOTP client. A router will also generally discard datagrams with the limited broadcast destination address. However, a

BOOTP relay agent will accept such datagrams from BOOTP clients on port 67.

The process carried out by a BOOTP relay agent on receiving a BOOTPREQUEST is as follows:

1. When the BOOTP relay agent receives a BOOTPREQUEST, it first checks the hops field to check the number of hops already completed in order to decide whether to forward the request. The threshold for the allowable number of hops is normally configurable.

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

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

#### 3.6.2 BOOTP considerations

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

### 3.7 Dynamic Host Configuration Protocol (DHCP)

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

The Dynamic Host Configuration Protocol (DHCP) provides a framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on the BOOTP protocol, adding the capability of automatic allocation of reusable network addresses and additional configuration options. For information regarding BOOTP, refer to 3.6, “Bootstrap Protocol (BOOTP)” on page 125. DHCP messages use UDP port 67, the BOOTP server's well-known port and UDP port 68, the BOOTP client's well-known port. DHCP participants can interoperate with BOOTP participants. See 3.7.8, “BOOTP and DHCP interoperability” on page 140 for further details.

DHCP consists of two components:

A protocol that delivers host-specific configuration parameters from a DHCP server to a host

A mechanism for the allocation of temporary or permanent network addresses to hosts

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

DHCP supports three mechanisms for IP address allocation:

Automatic allocation

DHCP assigns a permanent IP address to the host.

Dynamic allocation

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

Manual allocation

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

#### 3.7.1 The DHCP message format

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

*Figure 3-44 DHCP message format*

code

length

31

0

8 16 24

HWtype

hops

ction ID

transa

client IP address

your IP address

server IP address

seconds

flags field

router IP address

client hardware address

(16

bytes

)

server host name

(64

bytes

)

boot file name

(128

bytes

)

options

(312

bytes

)

Where:

|  |  |
| --- | --- |
| **Code** | Indicates a request or a reply:   1. Request 2. Reply |
| **HWtype** | The type of hardware, for example:  **1** Ethernet |
| **Length** | **6** IEEE 802 Networks    Refer to STD 2 – Assigned Internet Numbers for a complete list.  Hardware address length in bytes. |
| **Hops** | The client sets this to 0. It is incremented by a router that relays the request to another server and is used to identify loops. RFC 951 suggests that a value of 3 indicates a loop. | |
| **Transaction ID** | A random number used to match this boot request with the response it generates. | |
| **Seconds** | Set by the client. It is the elapsed time in seconds since the client started its boot process. | |
| **Flags field** | The most significant bit of the flags field is used as a broadcast flag. All other bits must be set to zero, and are reserved for future use. Normally, DHCP servers attempt to deliver DHCP messages directly to a client using unicast delivery. The destination address in the IP header is set to the DHCP *your IP address* and the MAC address is set to the DHCP *client hardware address*. If a host is unable to receive a unicast IP datagram until it knows its IP address, this broadcast bit must be set to indicate to the server that the DHCP reply must be sent as an IP and MAC broadcast.  Otherwise, this bit must be set to zero. | |
| **Client IP address** | Set by the client. Either its known IP address, or 0.0.0.0. | |
| **Your IP address** | Set by the server if the client IP address field was 0.0.0.0. | |
| **Server IP address** | Set by the server. | |
| **Router IP address** | This is the address of a BOOTP relay agent, *not* a general IP router to be used by the client. It is set by the forwarding agent when BOOTP forwarding is used (see 3.6.1, “BOOTP forwarding” on page 129). | |

##### Client hardware address

Set by the client. DHCP defines a client identifier option that is used for client identification. If this option is not used, the client is identified by its MAC address.

|  |  |
| --- | --- |
| **Server host name** | Optional server host name terminated by X'00'. |
| **Boot file name** | The client either leaves this null or specifies a generic name, such as router, indicating the type of boot file to be used. In a DHCPDISCOVER request, this is set to  null. The server returns a fully qualified directory path name in a DHCPOFFER request. The value is terminated by X'00'. |

**Options** The first four bytes of the options field of the DHCP

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

#### 3.7.2 DHCP message types

DHCP messages fall into one of the following categories:

DHCPDISCOVER: Broadcast by a client to find available DHCP servers.

DHCPOFFER: Response from a server to a DHCPDISCOVER and offering IP address and other parameters.

DHCPREQUEST: Message from a client to servers that does one of the following:

* Requests the parameters offered by one of the servers and declines all other offers.
* Verifies a previously allocated address after a system or network change (a reboot for example).
* Requests the extension of a lease on a particular address.

DHCPACK: Acknowledgement from server to client with parameters, including IP address.

DHCPNACK: Negative acknowledgement from server to client, indicating that the client's lease has expired or that a requested IP address is incorrect.

DHCPDECLINE: Message from client to server indicating that the offered address is already in use.

DHCPRELEASE: Message from client to server cancelling remainder of a lease and relinquishing network address.

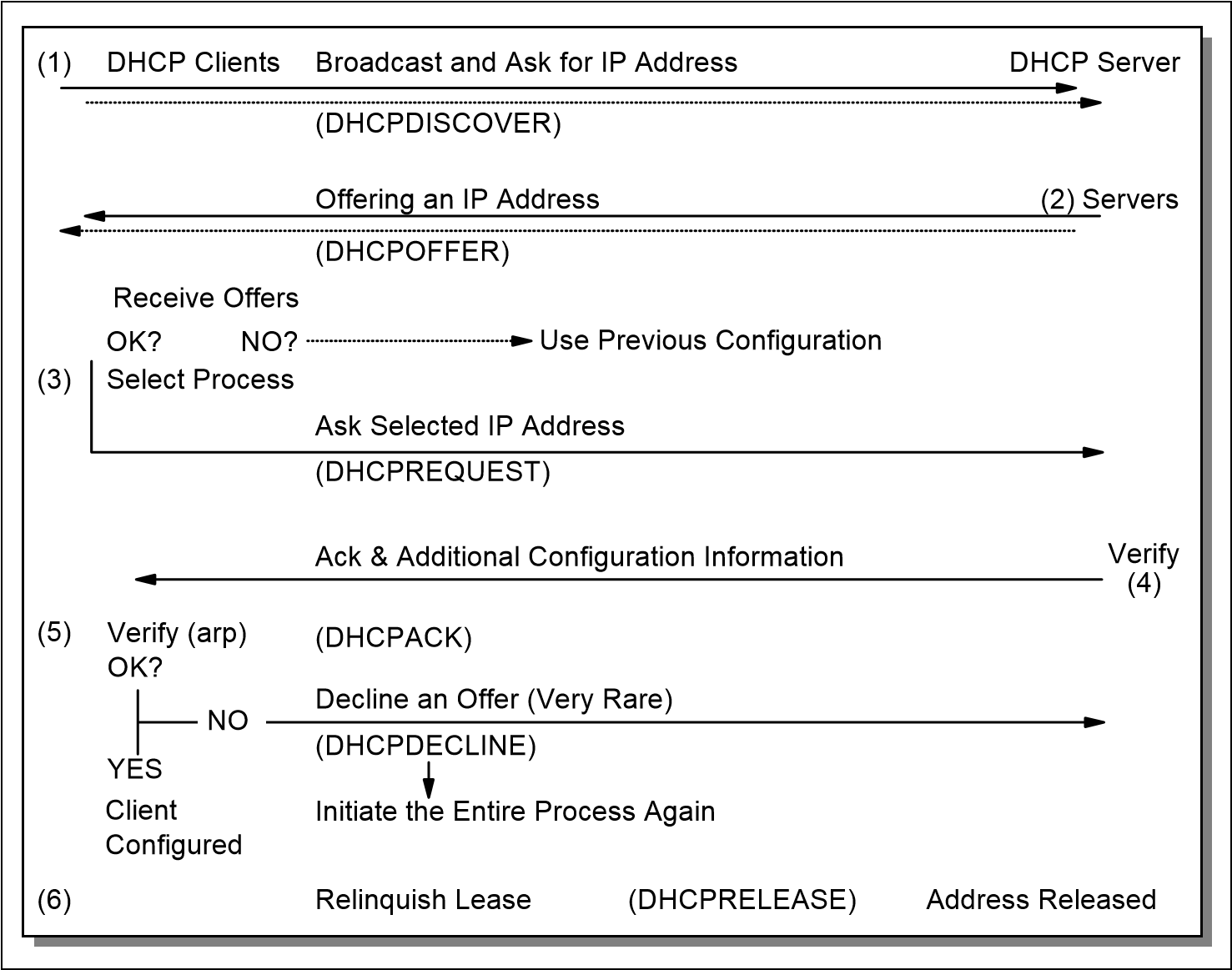
DHCPINFORM: Message from a client that already has an IP address (manually configured, for example), requesting further configuration parameters from the DHCP server.

#### 3.7.3 Allocating a new network address

This section describes the client/server interaction if the client does not know its network address. Assume that the DHCP server has a block of network

addresses from which it can satisfy requests for new addresses. Each server also maintains a database of allocated addresses and leases in permanent local storage.

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



*Figure 3-45 DHCP client and DHCP server interaction*

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

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

DHCPDISCOVER messages being received before the first client has

completed its configuration.

1. The client receives one or more DHCPOFFER messages from one or more servers. The client chooses one based on the configuration parameters offered and broadcasts a DHCPREQUEST message that includes the server identifier option to indicate which message it has selected and the requested IP address option taken from your IP address in the selected offer.
2. In the event that no offers are received, if the client has knowledge of a previous network address, the client can reuse that address if its lease is still valid until the lease expires.
3. The servers receive the DHCPREQUEST broadcast from the client. Those servers not selected by the DHCPREQUEST message use the message as

notification that the client has declined that server's offer. The server selected in the DHCPREQUEST message commits the binding for the client to persistent storage and responds with a DHCPACK message containing the configuration parameters for the requesting client. The combination of client hardware and assigned network address constitute a unique identifier for the client's lease and are used by both the client and server to identify a lease referred to in any DHCP messages. The your IP address field in the DHCPACK messages is filled in with the selected network address.

1. The client receives the DHCPACK message with configuration parameters. The client performs a final check on the parameters, for example, with ARP for allocated network address, and notes the duration of the lease and the lease identification cookie specified in the DHCPACK message. At this point, the client is configured.
2. If the client detects a problem with the parameters in the DHCPACK message (the address is already in use in the network, for example), the client sends a DHCPDECLINE message to the server and restarts the configuration process. The client should wait a minimum of ten seconds before restarting the configuration process to avoid excessive network traffic in case of looping. On receipt of a DHCPDECLINE, the server must mark the offered address as unavailable (and possibly inform the system administrator that there is a configuration problem).
3. If the client receives a DHCPNAK message, the client restarts the configuration process.
4. The client may choose to relinquish its lease on a network address by sending a DHCPRELEASE message to the server. The client identifies the lease to be released by including its network address and its hardware address.

#### 3.7.4 DHCP lease renewal process

This section describes the interaction between DHCP servers and clients that

have already been configured and the process that ensures lease expiration and renewal.

The process involves the following steps:

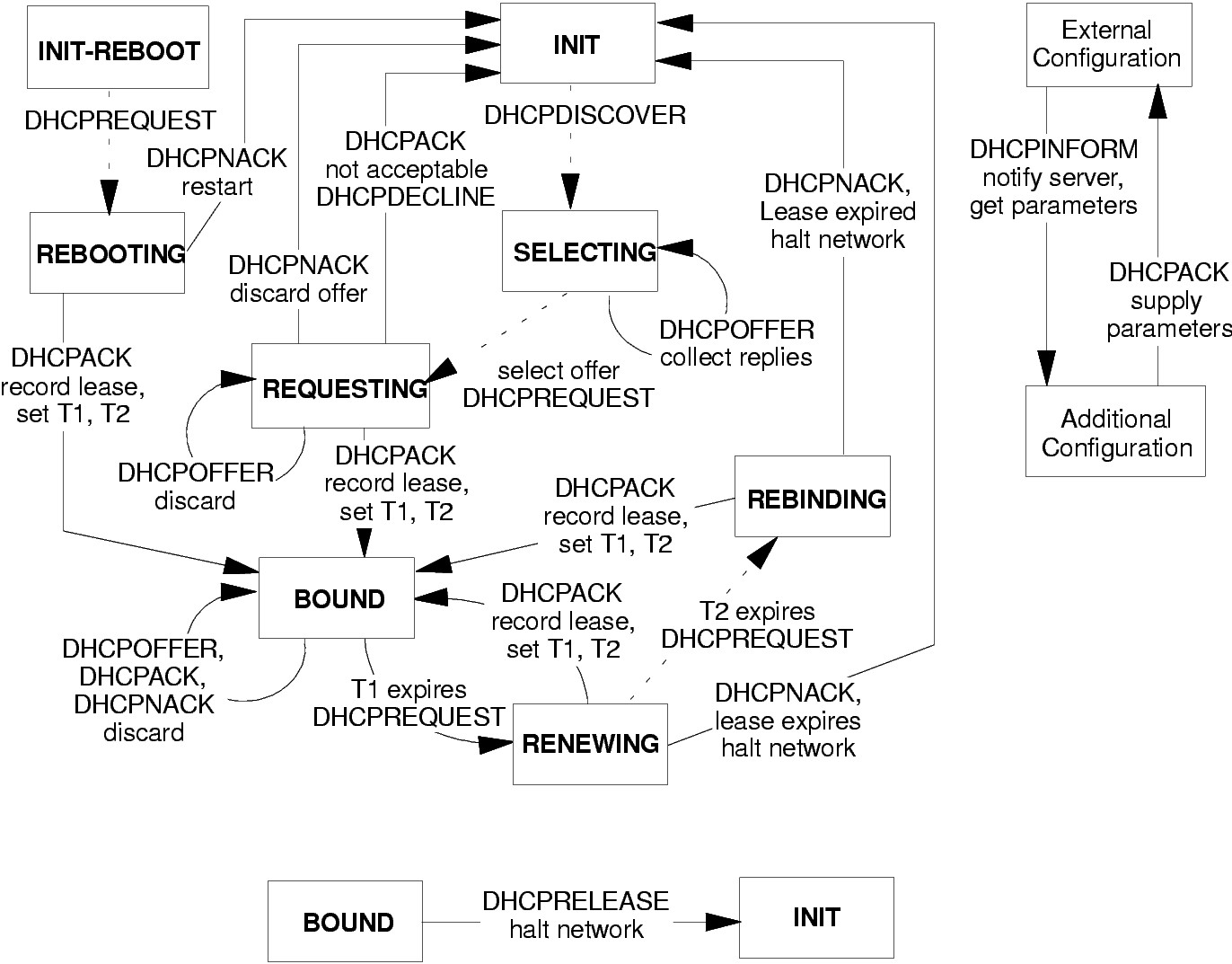
1. When a server sends the DHCPACK to a client with IP address and configuration parameters, it also registers the start of the lease time for that address. This lease time is passed to the client as one of the options in the DHCPACK message, together with two timer values, T1 and T2. The client is rightfully entitled to use the given address for the duration of the lease time. On applying the received configuration, the client also starts the timers T1 and T2. At this time, the client is in the BOUND state. Times T1 and T2 are options configurable by the server, but T1 must be less than T2, and T2 must be less than the lease time. According to RFC 2132, T1 defaults to (0.5 \* lease time) and T2 defaults to (0.875 \* lease time).
2. When timer T1 expires, the client will send a DHCPREQUEST (unicast) to the server that offered the address, asking to extend the lease for the given configuration. The client is now in the RENEWING state. The server usually responds with a DHCPACK message indicating the new lease time, and timers T1 and T2 are reset at the client accordingly. The server also resets its record of the lease time. In normal circumstances, an active client continually renews its lease in this way indefinitely, without the lease ever expiring.
3. If no DHCPACK is received until timer T2 expires, the client enters the

REBINDING state. It now broadcasts a DHCPREQUEST message to extend its lease. This request can be confirmed by a DHCPACK message from any DHCP server in the network.

1. If the client does not receive a DHCPACK message after its lease has expired, it has to stop using its current TCP/IP configuration. The client can then return to the INIT state, issuing a DHCPDISCOVER broadcast to try and obtain any valid address.

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

*Figure 3-46 DHCP client state and DHCP process*



#### 3.7.5 Reusing a previously allocated network address

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

1. The client broadcasts a DHCPREQUEST message on its local subnet. The DHCPREQUEST message includes the client's network address.
2. A server with knowledge of the client's configuration parameters responds with a DHCPACK message to the client (provided the lease is still current), renewing the lease at the same time.
3. If the client's lease has expired, the server with knowledge of the client responds with DHCPNACK.
4. The client receives the DHCPACK message with configuration parameters. The client performs a final check on the parameters and notes the duration of the lease and the lease identification cookie specified in the DHCPACK message. At this point, the client is configured and its T1 and T2 timers are reset.
5. If the client detects a problem with the parameters in the DHCPACK message, the client sends a DHCPDECLINE message to the server and restarts the configuration process by requesting a new network address. If the client receives a DHCPNAK message, it cannot reuse its remembered network address. It must instead request a new address by restarting the configuration process as described in 3.7.3, “Allocating a new network address” on page 134.

For further information, refer to the previously mentioned RFCs.

#### 3.7.6 Configuration parameters repository

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

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

#### 3.7.7 DHCP considerations

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

DHCP is built on UDP, which is inherently insecure. In normal operation, an unauthorized client can connect to a network and obtain a valid IP address and configuration. To prevent this, it is possible to preallocate IP addresses to particular MAC addresses (similar to BOOTP), but this increases the administration workload and removes the benefit of recycling of addresses. Unauthorized DHCP servers can also be set up, sending false and potentially disruptive information to clients.

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

#### 3.7.8 BOOTP and DHCP interoperability

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

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

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

### 3.8 RFCs relevant to this chapter

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

[RFC 791 – Internet Protocol (September 1981)](http://www.ietf.org/rfc/rfc791.txt)

[RFC 792 – Internet Control Message Protocol (September 1981)](http://www.ietf.org/rfc/rfc792.txt)

[RFC 826 – Ethernet Address Resolution Protocol: Or converting network](http://www.ietf.org/rfc/rfc826.txt)   [protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware (November 1982)](http://www.ietf.org/rfc/rfc826.txt)

[RFC 903 – A Reverse Address Resolution Protocol (June 1984)](http://www.ietf.org/rfc/rfc903.txt)

[RFC 906 – Bootstrap loading using TFTP (June 1984)](http://www.ietf.org/rfc/rfc906.txt)

[RFC919 – Broadcasting Internet Datagrams (October 1984)](http://www.ietf.org/rfc/rfc919.txt)

[RFC922 – Broadcasting Internet datagrams in the presence of subnets (October 1984)](http://www.ietf.org/rfc/rfc922.txt)

[RFC 925 – Multi-LAN address resolution (October 1984)](http://www.ietf.org/rfc/rfc925.txt)

[RFC 950 – Internet Standard Subnetting Procedure (August 1985)](http://www.ietf.org/rfc/rfc950.txt)

[RFC 951 – Bootstrap Protocol (September 1985)](http://www.ietf.org/rfc/rfc951.txt)

[RFC 1027 – Using ARP to implement transparent subnet gateways](http://www.ietf.org/rfc/rfc1027.txt)

[(October](http://www.ietf.org/rfc/rfc1027.txt) [1987)](http://www.ietf.org/rfc/rfc1027.txt)

[RFC 1112 – Host extensions for IP multicasting (August 1989)](http://www.ietf.org/rfc/rfc1112.txt)

[RFC 1122 – Requirements for Internet Hosts – Communication Layers (October 1989)](http://www.ietf.org/rfc/rfc1122.txt)

[RFC 1166 – Internet numbers (July 1990)](http://www.ietf.org/rfc/rfc1166.txt)

[RFC 1191 – Path MTU discovery (November 1990)](http://www.ietf.org/rfc/rfc1191.txt)

[RFC 1256 – ICMP Router Discovery Messages (September 1991)](http://www.ietf.org/rfc/rfc1256.txt)

[RFC 1349 – Type of Service in the Internet Protocol Suite (July 1992)](http://www.ietf.org/rfc/rfc1349.txt)

[RFC 1393 Traceroute Using an IP Option G (January 1993)](http://www.ietf.org/rfc/rfc1393.txt)

[RFC 1466 – Guidelines for Management of IP Address Space (May 1993)](http://www.ietf.org/rfc/rfc1466.txt)

[RFC 1518 – An Architecture for IP Address Allocation with CIDR](http://www.ietf.org/rfc/rfc1518.txt)

[(September](http://www.ietf.org/rfc/rfc1518.txt) [1993)](http://www.ietf.org/rfc/rfc1518.txt)

[RFC 1519 – Classless Inter-Domain Routing (CIDR): an Address Assignment (September 1993)](http://www.ietf.org/rfc/rfc1519.txt)

[RFC 1520 – Exchanging Routing Information Across Provider Boundaries in the CIDR Environment (September 1993)](http://www.ietf.org/rfc/rfc1520.txt)

[RFC 1542 – Clarifications and Extensions for the Bootstrap Protocol](http://www.ietf.org/rfc/rfc1524.txt)

[(October](http://www.ietf.org/rfc/rfc1524.txt) [1993)](http://www.ietf.org/rfc/rfc1524.txt)

[RFC 1788 – ICMP Domain Name Messages (April 1995)](http://www.ietf.org/rfc/rfc1788.txt)

[RFC 1812 – Requirements for IP Version 4 Routers (June 1995)](http://www.ietf.org/rfc/rfc1812.txt)

[RFC 1918 – Address Allocation for Private Internets (February 1996)](http://www.ietf.org/rfc/rfc1918.txt)

[RFC 2050 – Internet Registry IP Allocation Guidelines (November 1996)](http://www.ietf.org/rfc/rfc2050.txt)

[RFC 2131 – Dynamic Host Configuration Protocol (March 1997)](http://www.ietf.org/rfc/rfc2131.txt)

[RFC 2132 – DHCP Options and BOOTP Vendor Extensions (March 1997)](http://www.ietf.org/rfc/rfc2132.txt)

[RFC 2236 – Internet Group Management Protocol, Version 2](http://www.ietf.org/rfc/rfc2236.txt)

[(November](http://www.ietf.org/rfc/rfc2236.txt) [1997)](http://www.ietf.org/rfc/rfc2236.txt)

[RFC 2474 – Definition of the Differentiated Services Field (DS Field) in the](http://www.ietf.org/rfc/rfc2474.txt)

[IPv4 and IPv6 Headers (December 1998)](http://www.ietf.org/rfc/rfc2474.txt)

[RFC 2644 – Changing the Default for Directed Broadcasts in Router](http://www.ietf.org/rfc/rfc2644.txt)

[(August](http://www.ietf.org/rfc/rfc2644.txt) [1999)](http://www.ietf.org/rfc/rfc2644.txt)

[RFC 2663 – IP Network Address Translator (NAT) Terminology and](http://www.ietf.org/rfc/rfc2663.txt)

[Considerations (August 1999)](http://www.ietf.org/rfc/rfc2663.txt)

[RFC 3022 – Traditional IP Network Address Translator (Traditional NAT) (January 2001)](http://www.ietf.org/rfc/rfc3022.txt)

[RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP (September 2001)](http://www.ietf.org/rfc/rfc3168txt)

[RFC 3260 – New Terminology and Clarifications for Diffserv (April 2002)](http://www.ietf.org/rfc/rfc3260.txt)

[RFC 3330 – Special-Use IPv4 Addresses (September 2002)](http://www.ietf.org/rfc/rfc3330.txt)

[RFC 3396 – Encoding Long Options in the Dynamic Host Configuration](http://www.ietf.org/rfc/rfc3396.txt)

[Protocol (DHCPv4) (November 2002)](http://www.ietf.org/rfc/rfc3396.txt)

[RFC 3442 – The Classless Static Route Option for Dynamic Host](http://www.ietf.org/rfc/rfc3442.txt)

[Configuration Protocol (DHCP) version 4 (December 2002)](http://www.ietf.org/rfc/rfc3442.txt)

[RFC 3942 – Reclassifying Dynamic Host Configuration Protocol version 4](http://www.ietf.org/rfc/rfc3942txt)

[(DHCPv4) Options (November 2004)](http://www.ietf.org/rfc/rfc3942txt)

[RFC 4361 – Node-specific Client Identifiers for Dynamic Host Configuration](http://www.ietf.org/rfc/rfc4361.txt)

[Protocol Version Four (DHCPv4) (February 2006)](http://www.ietf.org/rfc/rfc4361.txt)

[RFC 4379 – Detecting Multi-Protocol Label Switched (MPLS) Data Plane](http://www.ietf.org/rfc/rfc479.txt)

[Failures (February 2006)](http://www.ietf.org/rfc/rfc479.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **4** | |

## Chapter 4. Transport layer protocols

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

User Datagram Protocol (UDP)

Transmission Control Protocol (TCP)

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

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

### 4.1 Ports and sockets

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

An application process is assigned a process identifier number (process ID), which is likely to be different each time that process is started.

Process IDs differ between operating system platforms, thus they are not uniform.

A server process can have multiple connections to multiple clients at a time, thus simple connection identifiers are not unique.The concept of ports and sockets provides a way to uniformly and uniquely identify connections and the programs and hosts that are engaged in them, irrespective of specific process IDs.

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

#### 4.1.1 Ports

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

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

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

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

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

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

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

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

#### 4.1.2 Sockets

The socket interface is one of several application programming interfaces to the communication protocols (see 11.2, “Application programming interfaces (APIs)” on page 410). Designed to be a generic communication programming interface, socket APIs were first introduced by 4.2 Berkeley Software Distribution (BSD). Although it has not been standardized, Berkeley socket API has become a de facto industry standard abstraction for network TCP/IP socket implementation.

Consider the following terminologies:

A *socket* is a special type of *file handle*, which is used by a process to request network services from the operating system.

A *socket address* is the triple:

<protocol, local-address, local port>

For example, in the TCP/IP (version 4) suite:

<tcp, 192.168.14.234, 8080>

A *conversation* is the communication link between two processes.

An *association* is the 5-tuple that completely specifies the two processes that comprise a connection:

<protocol, local-address, local-port, foreign-address, foreign-port>

In the TCP/IP (version 4) suite, the following could be a valid association:

<tcp, 192.168.14.234, 1500, 192.168.44, 22>

A *half-association* is either one of the following, which each specify half of a connection:

<protocol, local-address, local-process> Or:

<protocol, foreign-address, foreign-process>

The half-association is also called a *socket* or a *transport address*. That is, a socket is an endpoint for communication that can be named and addressed in a network.

Two processes communicate through TCP sockets. The socket model provides a process with a full-duplex byte stream connection to another process. The application need not concern itself with the management of this stream; these facilities are provided by TCP.

TCP uses the same port principle as UDP to provide multiplexing. Like UDP, TCP uses well-known and ephemeral ports. Each side of a TCP connection has a socket that can be identified by the triple <TCP, IP address, port number>. If two processes are communicating over TCP, they have a logical connection that is uniquely identifiable by the two sockets involved, that is, by the combination <TCP, local IP address, local port, remote IP address, remote port>. Server processes are able to manage multiple conversations through a single port. Refer to 11.2.1, “The socket API” on page 410 for more information about socket APIs.

### 4.2 User Datagram Protocol (UDP)

UDP is a standard protocol with STD number 6. UDP is described by RFC 768 – User Datagram Protocol. Its status is standard and almost every TCP/IP implementation intended for small data units transfer or those which can afford to lose a little amount of data (such as multimedia streaming) will include UDP.

UDP is basically an application interface to IP. It adds no reliability, flow-control, or error recovery to IP. It simply serves as a multiplexer/demultiplexer for sending and receiving datagrams, using ports to direct the datagrams, as shown in Figure 4-1. For a more detailed discussion of ports, refer to 4.1, “Ports and sockets” on page 144.

port b

process 2

port z

...

UDP - Port De-Multiplexing

IP

process n

process 1

port a

*Figure 4-1 UDP: Demultiplexing based on ports*

UDP provides a mechanism for one application to send a datagram to another. The UDP layer can be regarded as being extremely thin and is, consequently, very efficient, but it requires the application to take responsibility for error recovery and so on.

Applications sending datagrams to a host need to identify a target that is more specific than the IP address, because datagrams are normally directed to certain processes and not to the system as a whole. UDP provides this by using ports. We discuss the port concept in 4.1, “Ports and sockets” on page 144.

#### 4.2.1 UDP datagram format

Each UDP datagram is sent within a single IP datagram. Although, the IP datagram might be fragmented during transmission, the receiving IP implementation will reassemble it before presenting it to the UDP layer. All IP implementations are required to accept datagrams of 576 bytes, which means that, allowing for maximum-size IP header of 60 bytes, a UDP datagram of 516 bytes is acceptable to all implementations. Many implementations will accept larger datagrams, but this is not guaranteed.

The UDP datagram has an 8-byte header, as described in Figure 4-2 on page 148.

*Figure 4-2 UDP: Datagram format*

Destination Port

Source Port

Da

ta...

Checksum

Length

Where:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source Port** | | Indicates the port of the sending process. It is the port to which replies are addressed. | | | |
| **Destination Port** | | Specifies the port of the destination process on the destination host. | | | |
| **Length** | | The length (in bytes) of this user datagram, including the header. | | | |
| **Checksum** | | An optional 16-bit one's complement of the one's complement sum of a pseudo-IP header, the UDP header, and the UDP data. In Figure 4-3, we see a pseudo-IP header. It contains the source and destination IP addresses, the protocol, and the UDP length. | | | |
|  | | Source IP address |  |
|  | | Destination IP addr | ess |
| Zero | | Protocol | TCP Length |

*Figure 4-3 UDP: Pseudo-IP header*

The pseudo-IP header effectively extends the checksum to include the original (unfragmented) IP datagram.

#### 4.2.2 UDP application programming interface

The application interface offered by UDP is described in RFC 768. It provides for:

The creation of new receive ports

The receive operation that returns the data bytes and an indication of source port and source IP address

The send operation that has, as parameters, the data, source, and destination ports and addresses

The way this interface is implemented is left to the discretion of each vendor.

Be aware that UDP and IP do not provide guaranteed delivery, flow-control, or error recovery, so these must be provided by the application.

Standard applications using UDP include:

Trivial File Transfer Protocol (see 14.2, “Trivial File Transfer Protocol (TFTP)” on page 529).

Domain Name System name server (see 12.2, “Dynamic Domain Name System” on page 453).

Remote Procedure Call, used by the Network File System (see both 11.2.2, “Remote Procedure Call (RPC)” on page 415 and 14.4, “Network File System (NFS)” on page 538).

Simple Network Management Protocol (see 17.1, “The Simple Network Management Protocol (SNMP)” on page 624).

Lightweight Directory Access Protocol (see 12.4, “Lightweight Directory Access Protocol (LDAP)” on page 459).

### 4.3 Transmission Control Protocol (TCP)

TCP is a standard protocol with STD number 7. TCP is described by RFC 793 – Transmission Control Protocol. Its status is standard, and in practice, every TCP/IP implementation that is not used exclusively for routing will include TCP.

TCP provides considerably more facilities for applications than UDP. Specifically, this includes error recovery, flow control, and reliability. TCP is a *connection-oriented* protocol, unlike UDP, which is *connectionless*. Most of the user application protocols, such as Telnet and FTP, use TCP. The two processes communicate with each other over a TCP connection (InterProcess Communication, or IPC), as shown in Figure 4-4. In the figure, processes 1 and 2 communicate over a TCP connection carried by IP datagrams. See 4.1, “Ports and sockets” on page 144 for more details about ports and sockets.

port m

reliable

TCP connection

...

TCP

IP

...

port n

process 2

...

TCP

IP

...

host A

unreliable

IP datagrams

process 1

host B

*Figure 4-4 TCP: Connection between processes*

#### 4.3.1 TCP concept

As noted earlier, the primary purpose of TCP is to provide a reliable logical circuit or connection service between pairs of processes. It does *not* assume reliability from the lower-level protocols (such as IP), so TCP must guarantee this itself.

TCP can be characterized by the following facilities it provides for the applications using it:

Stream data transfer: From the application's viewpoint, TCP transfers a contiguous stream of bytes through the network. The application does not have to bother with chopping the data into basic blocks or datagrams. TCP does this by grouping the bytes into TCP segments, which are passed to the IP layer for transmission to the destination. Also, TCP itself decides how to segment the data, and it can forward the data at its own convenience. Sometimes, an application needs to be sure that all the data passed to TCP has actually been transmitted to the destination. For that reason, a push function is defined. It will push all remaining TCP segments still in storage to

the destination host. The normal close connection function also pushes the data to the destination.

Reliability: TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP layer. If the ACK is not received within a timeout interval, the data is retransmitted. Because the data is transmitted in blocks (TCP segments), only the sequence number of the first data byte in the segment is sent to the destination host.

The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.

Flow control: The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive (beyond the last received TCP segment) without causing overrun and overflow in its internal buffers. This is sent in the ACK in the form of the highest sequence number it can receive without problems. This mechanism is also referred to as a window-mechanism, and we discuss it in more detail later in this chapter.

Multiplexing: Achieved through the use of ports, just as with UDP.

Logical connections: The reliability and flow control mechanisms described here require that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.

Full duplex: TCP provides for concurrent data streams in both directions.

##### The window principle

A simple transport protocol might use the following principle: send a packet and then wait for an acknowledgment from the receiver before sending the next packet. If the ACK is not received within a certain amount of time, retransmit the packet. See Figure 4-5 for more details.

|  |  |
| --- | --- |
| Sender | Receiver |
| Send packet 1  Receive ACK  Send packet 2 | Receive packet 1 and reply with an ACK 1 |

*Figure 4-5 TCP: The window principle*

Although this mechanism ensures reliability, it only uses a part of the available network bandwidth.

Now, consider a protocol where the sender groups its packets to be transmitted, as in Figure 4-6, and uses the following rules:

The sender can send all packets within the window without receiving an ACK, but must start a timeout timer for each of them.

The receiver must acknowledge each packet received, indicating the sequence number of the last well-received packet.

The sender slides the window on each ACK received.

*Figure 4-6 TCP: Message packets*

window

packets

3

4

5

6

7

8

9

...

2

1

As shown in Figure 4-7, the sender can transmit packets 1 to 5 without waiting for any acknowledgment.

*Figure 4-7 TCP: Window principle*

Sender

Network

Send packet 1

Send packet 2

Send packet 3

Send packet 4

Send packet 5

ACK for packet 1 received

ACK 1

As shown in Figure 4-8, at the moment the sender receives ACK 1

(acknowledgment for packet 1), it can slide its window one packet to the right.

|  |  |
| --- | --- |
| Sender | Receiver |
| Send packet 1  Receive ACK  Send packet 2 | Receive packet 1 and reply with an ACK 1 |

*Figure 4-8 TCP: Message packets*

At this point, the sender can also transmit packet 6.

Imagine some special cases:

Packet 2 gets lost: The sender will not receive ACK 2, so its window will remain in position 1 (as in Figure 4-8 on page 153). In fact, because the receiver did not receive packet 2, it will acknowledge packets 3, 4, and 5 with an ACK 1, because packet 1 was the last one received in sequence. At the sender's side, eventually a timeout will occur for packet 2 and it will be retransmitted. Note that reception of this packet by the receiver will generate ACK 5, because it has now successfully received all packets 1 to 5, and the sender's window will slide four positions upon receiving this ACK 5.

Packet 2 did arrive, but the acknowledgment gets lost: The sender does not receive ACK 2, but will receive ACK 3. ACK 3 is an acknowledgment for *all* packets up to 3 (including packet 2) and the sender can now slide its window to packet 4.

This window mechanism ensures:

Reliable transmission.

Better use of the network bandwidth (better throughput).

Flow-control, because the receiver can delay replying to a packet with an acknowledgment, knowing its free buffers are available and the window size of the communication.

##### The window principle applied to TCP

The previously discussed window principle is used in TCP, but with a few differences:

Because TCP provides a byte-stream connection, sequence numbers are assigned to each byte in the stream. TCP divides this contiguous byte stream into TCP segments to transmit them. The window principle is used at the byte level, that is, the segments sent and ACKs received will carry byte-sequence numbers and the window size is expressed as a number of bytes, rather than a number of packets.

The window size is determined by the receiver when the connection is established and is variable during the data transfer. Each ACK message will include the window size that the receiver is ready to deal with at that particular time.

The sender's data stream can now be seen as follows in Figure 4-9.

5

4

2

1

...

1

14

13

12

11

10

9

8

7

3

6

5

A

B

C

D

window (size expressed in bytes)

bytes

*Figure 4-9 TCP: Window principle applied to TCP*

Where:

1. Bytes that are transmitted and have been acknowledged
2. Bytes that are sent but not yet acknowledged
3. Bytes that can be sent without waiting for any acknowledgment
4. Bytes that cannot be sent yet

Remember that TCP will block bytes into segments, and a TCP segment only carries the sequence number of the first byte in the segment.

##### TCP segment format

Figure 4-10 shows the TCP segment format.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | | 1 | | | 2 | | | | | | 3 |
| 0 1 2 3 4 5 6 7 8 9 | | 0 1 2 3 4 5 6 7 8 9 | | | 0 1 2 3 4 5 6 7 8 9 | | | | | | 0 1 2 3 4 5 6 7 8 9 |
| Source Port | | | | | Destination Port | | | | | | |
| Sequence Number | | | | | | | | | | | |
| Acknowledgment Number | | | | | | | | | | | |
| Data  Offset | Reserved | | U  R  G | A  C  K | | P  S  H | R  S  T | S  Y N | F  I  N | Window | |
| Checksum | | | | | | | | Urgent Pointer | | | |
| Options …|… Padding | | | | | | | | | | | |
| Data Bytes | | | | | | | | | | | |

*Figure 4-10 TCP: Segment format*

Where:

**Source Port** The 16-bit source port number, used by the receiver to

reply.

**Destination Port** The 16-bit destination port number.

**Sequence Number** The sequence number of the first data byte in this segment. If the SYN control bit is set, the sequence number is the initial sequence number (n) and the first data byte is n+1.

###### Acknowledgment Number

If the ACK control bit is set, this field contains the value of the next sequence number that the receiver is expecting to receive.

|  |  |  |
| --- | --- | --- |
| **Data Offset** | The number of 32-bit words in the TCP header. It indicates where the data begins. | |
| **Reserved** | Six bits reserved for future use; must be zero. | |
| **URG** | Indicates that the urgent pointer field is significant in this segment. | |
| **ACK**  **PSH**  **RST** | Indicates that the acknowledgment field is significant in this segment.  Push function.  Resets the connection. | |
| **SYN** | Synchronizes the sequence numbers. |
| **FIN** | No more data from sender. |
| **Window** | Used in ACK segments. It specifies the number of data bytes, beginning with the one indicated in the acknowledgment number field that the receiver (the sender of this segment) is willing to accept. |
| **Checksum** | The 16-bit one's complement of the one's complement sum of all 16-bit words in a pseudo-header, the TCP header, and the TCP data. While computing the checksum, the checksum field itself is considered zero. |

The pseudo-header is the same as that used by UDP for calculating the checksum. It is a pseudo-IP-header, only used for the checksum calculation, with the format shown in Figure 4-11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Source IP address |  |  |
|  |  | Destination IP addr | ess |
|  | Zero | Protocol | TCP Length |

*Figure 4-11 TCP: Pseudo-IP header*

**Urgent Pointer** Points to the first data octet following the urgent data. Only significant when the URG control bit is set.

**Options** Just as in the case of IP datagram options, options can be

either:

* A single byte containing the option number
* A variable length option in the following format as shown in Figure 4-12

3

3

option

length

option data...

*Figure 4-12 TCP: IP datagram option, variable length option*

There are currently seven options defined, as shown in Table 4-1.

*Table 4-1 TCP: IP datagram options*

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

###### Maximum segment size option

This option is only used during the establishment of the connection (SYN control bit set) and is sent from the side that is to receive data to indicate the maximum segment length it can handle. If this option is not used, any segment size is allowed. See Figure 4-13 for more details.

3

3

2

max.

seg size

4

*Figure 4-13 TCP: Maximum segment size*

###### Window scale option

This option is not mandatory. Both sides must send the Window scale option in their SYN segments to enable windows scaling in their direction. The Window scale expands the definition of the TCP window to 32 bits. It defines the 32-bit window size by using scale factor in the SYN segment over standard 16-bit window size. The receiver rebuilds the 32-bit window size by using the 16-bit window size and scale factor. This option is determined while handshaking. There is no way to change it after the connection has been established. See Figure 4-14 for more details.

3

3

shift.cnt

*Figure 4-14 TCP: Window scale option*

###### SACK-permitted option

This option is set when selective acknowledgment is used in that TCP connection. See Figure 4-15 for details.

4

2

*Figure 4-15 TCP: SACK-permitted option*

**SACK option** Selective Acknowledgment (SACK) allows the receiver to inform the sender about all the segments that are received successfully. Therefore, the sender will only send the segments that got lost. If the number of the segments that have been lost since the last SACK is too large, the SACK option will be too large. As a result, the number of blocks that can be reported by the SACK option is limited to four. To reduce this, the SACK option should be used for the most recent received data. See Figure 4-16 for more details.

*Figure 4-16 TCP: SACK option*

Length

5

Left Edge of 1st Block

Right Edge of 1st Block

-

- - - - -

-

Left Edge of Nth Block

Right Edge of Nth Bloc

k

/

/

/

/

**Timestamps option** The timestamps option sends a time stamp value that

indicates the current value of the time stamp clock of the TCP sending the option. The Timestamp Echo Value can only be used if the ACK bit is set in the TCP header. See Figure 4-17 for more details.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | | |
|  | |  |  |  |  | |
|  | 8 | 10 | TS Valve | TS Echo Reply |  |  |
|  |  |  |  |  |
| 1 | 1 | 4 | 4 | |

*Figure 4-17 TCP: Timestamps option*

**Padding** All zero bytes are used to fill up the TCP header to a total length that is a multiple of 32 bits.

##### Acknowledgments and retransmissions

TCP sends data in variable length segments. Sequence numbers are based on a byte count. Acknowledgments specify the sequence number of the next byte that the receiver expects to receive.

Consider that a segment gets lost or corrupted. In this case, the receiver will acknowledge all further well-received segments with an acknowledgment referring to the first byte of the missing packet. The sender will stop transmitting when it has sent all the bytes in the window. Eventually, a timeout will occur and the missing segment will be retransmitted.

Figure 4-18 illustrates and example where a window size of 1500 bytes and segments of 500 bytes are used.

Sender

Receiver

Segment 1 (seq. 1000)

Receives 1000, sends ACK 1500

Segment 2 (seq. 1500)

\\\

gets lost

Segment 3 (seq. 2000)

Receives the ACK 1500,

which slides window

Segment 4 (seq. 2500)

Receives one of the frames

and replies with ACK 1500

(

receiver is still expecting

byte 1500)

window size reached,

waiting for ACK

Receives the ACK 1500,

which does not slide the

window

Timeout for Segment 2

Retransmission

*Figure 4-18 TCP: Acknowledgment and retransmission process*

A problem now arises, because the sender does know that segment 2 is lost or corrupted, but does not know anything about segments 3 and 4. The sender should at least retransmit segment 2, but it could also retransmit segments 3 and 4 (because they are within the current window). It is possible that:

Segment 3 has been received, and we do not know about segment 4. It might be received, but ACK did not reach us yet, or it might be lost.

Segment 3 was lost, and we received the ACK 1500 on the reception of segment 4.

Each TCP implementation is free to react to a timeout as those implementing it want. It can retransmit only segment 2, but in the second case, we will be waiting again until segment 3 times out. In this case, we lose all of the throughput advantages of the window mechanism. Or TCP might immediately resend all of the segments in the current window.

Whatever the choice, maximal throughput is lost. This is because the ACK does not contain a second acknowledgment sequence number indicating the actual frame received.

###### Variable timeout intervals

Each TCP should implement an algorithm to adapt the timeout values to be used for the round trip time of the segments. To do this, TCP records the time at which a segment was sent, and the time at which the ACK is received. A weighted average is calculated over several of these round trip times, to be used as a timeout value for the next segment or segments to be sent.

This is an important feature, because delays can vary in IP network, depending on multiple factors, such as the load of an intermediate low-speed network or the saturation of an intermediate IP gateway.

##### Establishing a TCP connection

Before any data can be transferred, a connection has to be established between the two processes. One of the processes (usually the server) issues a *passive OPEN* call, the other an *active OPEN* call. The passive OPEN call remains dormant until another process tries to connect to it by an active OPEN.

As shown in Figure 4-19, in the network, three TCP segments are exchanged.

Initiating

)

client

(

TCP Layer

Listening

(

server

)

TCP Layer

1)

SYN SEQ:999 ACK

:

2)

SYN ACK SEQ:4999 ACK

:1000

3)

ACK SEQ:1000 ACK

:5000

*Figure 4-19 TCP: Connection establishment*

This whole process is known as a three-way handshake. Note that the exchanged TCP segments include the initial sequence numbers from both sides, to be used on subsequent data transfers.

Closing the connection is done implicitly by sending a TCP segment with the FIN bit (no more data) set. Because the connection is full-duplex (that is, there are two independent data streams, one in each direction), the FIN segment only closes the data transfer in one direction. The other process will now send the remaining data it still has to transmit and also ends with a TCP segment where the FIN bit is set. The connection is deleted (status information on both sides) after the data stream is closed in both directions.

The following is a list of the different states of a TCP connection:

LISTEN: Awaiting a connection request from another TCP layer.

SYN-SENT: A SYN has been sent, and TCP is awaiting the response SYN.

SYN-RECEIVED: A SYN has been received, a SYN has been sent, and TCP is awaiting an ACK.

ESTABLISHED: The three-way handshake has been completed.

FIN-WAIT-1: The local application has issued a CLOSE. TCP has sent a FIN, and is awaiting an ACK or a FIN.

FIN-WAIT-2: A FIN has been sent, and an ACK received. TCP is awaiting a FIN from the remote TCP layer.

CLOSE-WAIT: TCP has received a FIN, and has sent an ACK. It is awaiting a close request from the local application before sending a FIN.

CLOSING: A FIN has been sent, a FIN has been received, and an ACK has been sent. TCP is awaiting an ACK for the FIN that was sent.

LAST-ACK: A FIN has been received, and an ACK and a FIN have been sent.

TCP is awaiting an ACK.

TIME-WAIT: FINs have been received and ACK’d, and TCP is waiting two MSLs to remove the connection from the table.

CLOSED: Imaginary, this indicates that a connection has been removed from the connection table.

#### 4.3.2 TCP application programming interface

The TCP application programming interface is not fully defined. Only some base functions it should provide are described in RFC 793 – Transmission Control Protocol. As is the case with most RFCs in the TCP/IP protocol suite, a great degree of freedom is left to the implementers, thereby allowing for optimal operating system-dependent implementations, resulting in better efficiency and greater throughput.

The following function calls are described in the RFC:

Open: To establish a connection takes several parameters, such as:

* Active/Passive
* Foreign socket
* Local port number
* Timeout value (optional)

This returns a local connection name, which is used to reference this particular connection in all other functions.

Send: Causes data in a referenced user buffer to be sent over the connection. Can optionally set the URGENT flag or the PUSH flag.

Receive: Copies incoming TCP data to a user buffer.

Close: Closes the connection; causes a push of all remaining data and a TCP segment with FIN flag set.

Status: An implementation-dependent call that can return information, such as:

* Local and foreign socket
* Send and receive window sizes
* Connection state
* Local connection name

Abort: Causes all pending Send and Receive operations to be aborted, and a RESET to be sent to the foreign TCP.

For full details, see RFC 793 – Transmission Control Protocol.

#### 4.3.3 TCP congestion control algorithms

One big difference between TCP and UDP is the congestion control algorithm. The TCP congestion algorithm prevents a sender from overrunning the capacity of the network (for example, slower WAN links). TCP can adapt the sender's rate

to network capacity and attempt to avoid potential congestion situations. In order to understand the difference between TCP and UDP, understanding basic TCP congestion control algorithms is very helpful.

Several congestion control enhancements have been added and suggested to TCP over the years. This is still an active and ongoing research area, but modern implementations of TCP contain four intertwined algorithms as basic Internet standards:

Slow start

Congestion avoidance

Fast retransmit

Fast recovery

##### Slow start

Old implementations of TCP start a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. Although this is OK when the two hosts are on the same LAN, if there are routers and slower links between the sender and the receiver, problems can arise. Some intermediate routers cannot handle it, packets get dropped, and retransmission results and performance is degraded.

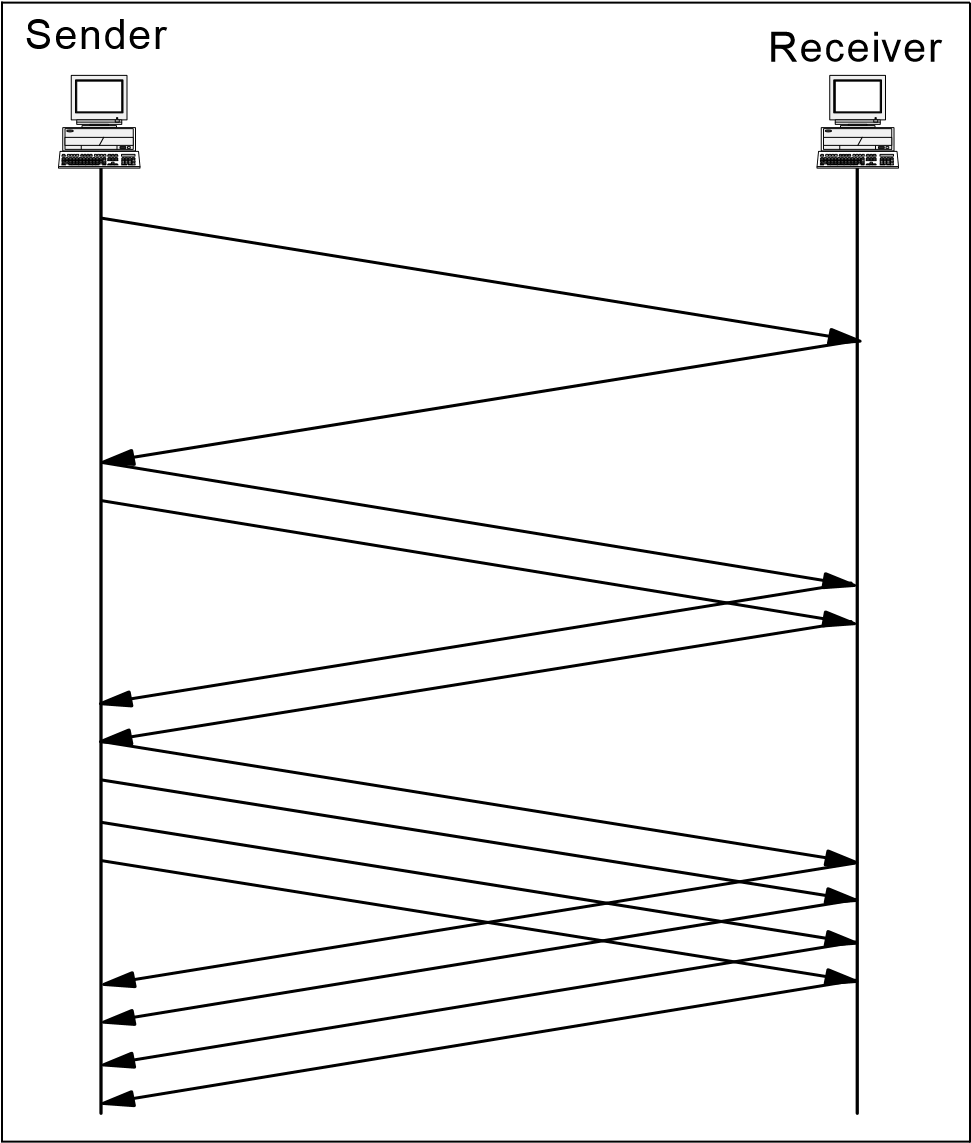
The algorithm to avoid this is called slow start. It operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgments are returned by the other end. Slow start adds another window to the sender's TCP: the congestion window, called cwnd. When a new connection is established with a host on another network, the congestion window is initialized to one segment (for example, the segment size announced by the other end, or the default, typically 536 or 512).

**Note:** Congestion control is defined in RFC 2581. Additionally, RFC 3390 updates RFC 2581 such that TCP implementations can initialize the congestion window to between two and four segments, with an upper limit of 4 K.

Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit the lower value of the congestion window or the advertised window. The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for this connection.

The sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth, although it is not exactly exponential, because the receiver might delay its ACKs, typically sending one ACK for every two segments that it receives.

At some point, the capacity of the IP network (for example, slower WAN links) can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has gotten too large. See Figure 4-20 for an overview of slow start in action.



*Figure 4-20 TCP: Slow start in action*

##### Congestion avoidance

The assumption of the algorithm is that packet loss caused by damage is very small (much less than 1%). Therefore, the loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss:

A timeout occurs.

Duplicate ACKs are received.

Congestion avoidance and slow start are independent algorithms with different objectives. But when congestion occurs, TCP must slow down its transmission rate of packets into the network and invoke slow start to get things going again. In practice, they are implemented together.

Congestion avoidance and slow start require that two variables be maintained for each connection:

A congestion window, cwnd A slow start threshold size, ssthresh

The combined algorithm operates as follows:

1. Initialization for a given connection sets cwnd to one segment and ssthresh to 65535 bytes.
2. The TCP output routine never sends more than the lower value of cwnd or the receiver's advertised window.
3. When congestion occurs (timeout or duplicate ACK), one-half of the current window size is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment.
4. When new data is acknowledged by the other end, increase cwnd, but the way it increases depends on whether TCP is performing slow start or congestion avoidance. If cwnd is less than or equal to ssthresh, TCP is in slow start; otherwise, TCP is performing congestion avoidance.

Slow start continues until TCP is halfway to where it was when congestion occurred (since it recorded half of the window size that caused the problem in step 2), and then congestion avoidance takes over. Slow start has cwnd begin at one segment, and incremented by one segment every time an ACK is received. As mentioned earlier, this opens the window exponentially: send one segment, then two, then four, and so on.

Congestion avoidance dictates that cwnd be incremented by segsize\*segsize/cwnd each time an ACK is received, where segsize is the segment size and cwnd is maintained in bytes. This is a linear growth of cwnd, compared to slow start's exponential growth. The increase in cwnd should be at most one segment each round-trip time (regardless of how many ACKs are received in that round-trip time), while slow start increments cwnd by the number of ACKs received in a round-trip time. Many implementations incorrectly add a small fraction of the segment size (typically the segment size divided by 8) during congestion avoidance. This is wrong and should not be emulated in future releases. See Figure 4-21 for an example of TCP slow start and congestion avoidance in action.

Round Trip Times

CWND

cwnd

ssthresh

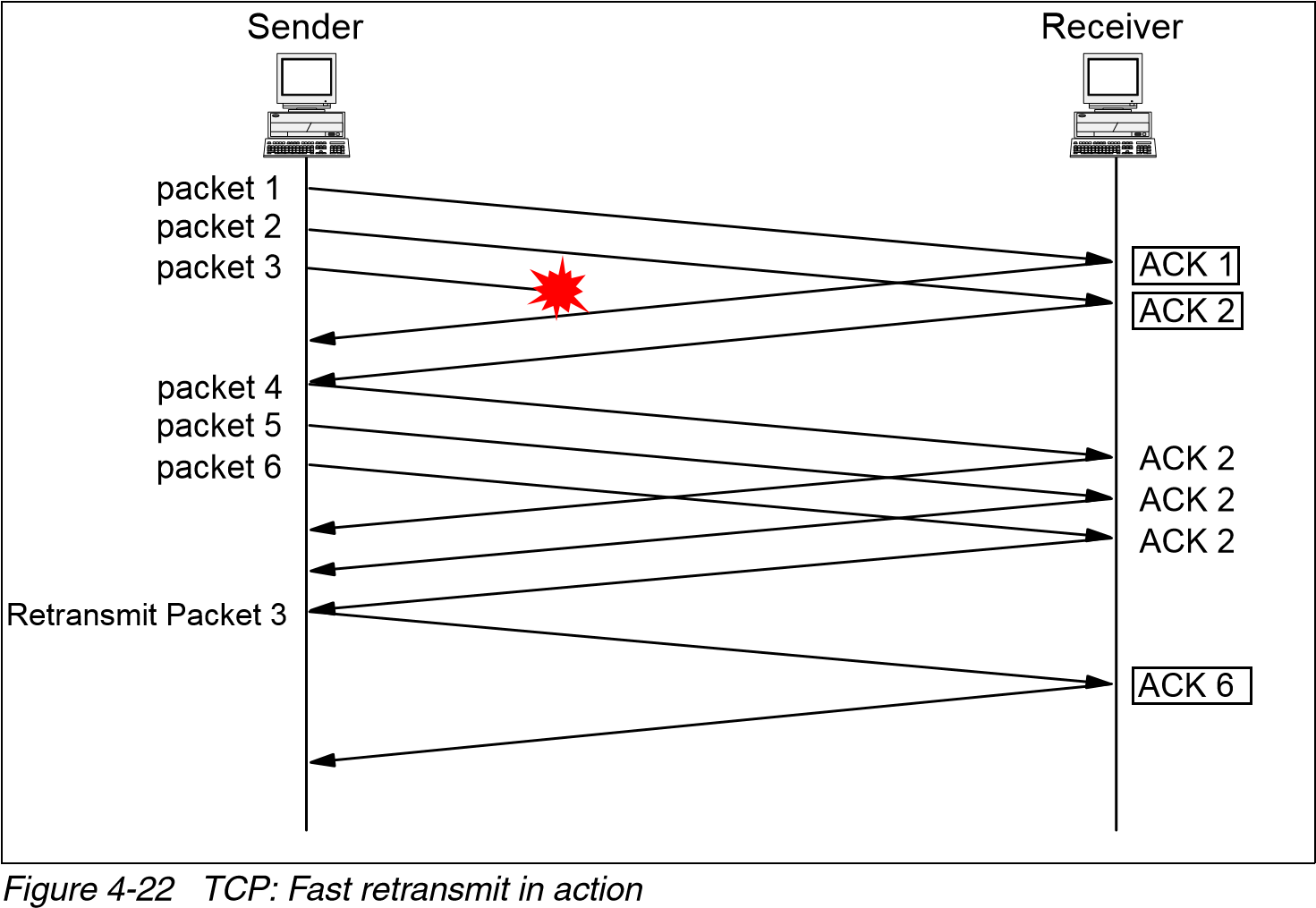
*Figure 4-21 TCP: Slow start and congestion avoidance behavior in action*

##### Fast retransmit

Fast retransmit avoids having TCP wait for a timeout to resend lost segments.

Modifications to the congestion avoidance algorithm were proposed in 1990. Before describing the change, realize that TCP can generate an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order and to tell it what sequence number is expected.

Because TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. See Figure 4-22 for an overview of TCP fast retransmit in action.



##### Fast recovery

After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start, is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows.

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Because the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start. The fast retransmit and fast recovery algorithms are usually implemented together as follows:

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus three times the segment size. This inflates the congestion window by the number of segments that have left the network and the other end has cached (3).
2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.
3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK is the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK acknowledges all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, because TCP is down to one-half the rate it was at when the packet was lost.

### 4.4 RFCs relevant to this chapter

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

RFC 761 – DoD standard Transmission Control Protocol (January 1980)

RFC 768 – User Datagram Protocol (August 1980)

RFC 793 – Updated by RFC 3168 - The Addition of Explicit Congestion

Notification (ECN) to IP (September 2001)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **5** | |

## Chapter 5. Routing protocols

This chapter provides an overview of IP routing and discusses the various routing protocols used.

One of the basic functions provided by the IP protocol is the ability to form connections between different physical networks. A system that performs this function is called an *IP router*. This type of device attaches to two or more physical networks and forwards datagrams between the networks.

When sending data to a remote destination, a host passes datagrams to a local router. The router forwards the datagrams toward the final destination. They travel from one router to another until they reach a router connected to the destination’s LAN segment. Each router along the end-to-end path selects the *next hop* device used to reach the destination. The next hop represents the next device along the path to reach the destination. It is located on a physical network connected to this intermediate system. Because this physical network differs from the one on which the system originally received the datagram, the intermediate host has *forwarded* (that is, routed) the IP datagram from one physical network to another.

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

Figure 5-1 shows an environment where Host C is positioned to forward packets between network X and network Y.

*Figure 5-1 IP routing operations*

Application

TCP

IP

Interface X

Host A

Application

TCP

IP

Interface Y

Host

B

Host C Acting as

Router

IP Routing

Interface X

Interface Y

Network Y

Network X

The IP routing table in each device is used to forward packets between network segments. The basic table contains information about a router’s locally connected networks. The configuration of the device can be extended to contain information detailing remote networks. This information provides a more complete view of the overall environment.

A robust routing protocol provides the ability to dynamically build and manage the information in the IP routing table. As network topology changes occur, the routing tables are updated with minimal or no manual intervention. This chapter details several IP routing protocols and how each protocol manages this information.

**Note:** In other sections of this book, the position of each protocol within the layered model of the OSI protocol stack is shown. The routing function is included as part of the internetwork layer. However, the primary function of a routing protocol is to exchange routing information with other routers. In this respect, routing protocols behave more like an application protocol. Therefore, this chapter makes no attempt to represent the position of these protocols within the overall protocol stack.

**Note:** Early IP routing documentation often referred to an IP router as an *IP gateway.*

### 5.1 Autonomous systems

The definition of an autonomous system (AS) is integral to understanding the function and scope of a routing protocol. An AS is defined as a logical portion of

a larger IP network. An AS normally consists of an internetwork within an organization. It is administered by a single management authority. As shown in Figure 5-2, an AS can connect to other autonomous systems managed by the same organization. Alternatively, it can connect to other public or private networks.

Router

Router

IGPs

Autonomous System A

Router

IGPs

Autonomous System C

Router

Router

Router

Single Management Authority

EGP

Router

Router

Router

Autonomous System B

IGPs

Internet

*Figure 5-2 Autonomous systems*

Some routing protocols are used to determine routing paths within an AS. Others are used to interconnect a set of autonomous systems:

Interior Gateway Protocols (IGPs): Interior Gateway Protocols allow routers to exchange information within an AS. Examples of these protocols are Open

Short Path First (OSPF) and Routing Information Protocol (RIP).

Exterior Gateway Protocols (EGPs): Exterior Gateway Protocols allow the exchange of summary information between autonomous systems. An example of this type of routing protocol is Border Gateway Protocol (BGP).

Figure 5-2 on page 173 depicts the interaction between Interior and Exterior Gateway Protocols. It shows the Interior Gateway Protocols used to maintain routing information within each AS. The figure also shows the Exterior Gateway Protocols maintaining the routing information between autonomous systems.

Within an AS, multiple interior routing processes can be used. When this occurs, the AS must appear to other autonomous systems as having a single coherent interior routing plan. The AS must present a consistent view of the internal destinations.

### 5.2 Types of IP routing and IP routing algorithms

Routing algorithms build and maintain the IP routing table on a device. There are two primary methods used to build the routing table:

Static routing: Static routing uses preprogrammed definitions representing paths through the network.

Dynamic routing: Dynamic routing algorithms allow routers to automatically discover and maintain awareness of the paths through the network. This automatic discovery can use a number of currently available dynamic routing protocols. The difference between these protocols is the way they discover and calculate new routes to destination networks. They can be classified into four broad categories:

* Distance vector protocols
* Link state protocols
* Path vector protocols
* Hybrid protocols

The remainder of this section describes the operation of each algorithm.

There are several reasons for the multiplicity of protocols:

Routing within a network and routing between networks typically have different requirements for security, stability, and scalability. Different routing protocols have been developed to address these requirements.

New protocols have been developed to address the observed deficiencies in established protocols.

Different-sized networks can use different routing algorithms. Small to medium-sized networks often use routing protocols that reflect the simplicity of the environment.

However, these protocols do not scale to support large, interconnected networks. More complex routing algorithms are required to support these environments.

#### 5.2.1 Static routing

Static routing is manually performed by the network administrator. The administrator is responsible for discovering and propagating routes through the network. These definitions are manually programmed in every routing device in the environment.

After a device has been configured, it simply forwards packets out the predetermined ports. There is no communication between routers regarding the current topology of the network.

In small networks with minimal redundancy, this process is relatively simple to administer. However, there are several disadvantages to this approach for maintaining IP routing tables:

Static routes require a considerable amount of coordination and maintenance in non-trivial network environments.

Static routes cannot dynamically adapt to the current operational state of the network. If a destination subnetwork becomes unreachable, the static routes pointing to that network remain in the routing table. Traffic continues to be forwarded toward that destination. Unless the network administrator updates the static routes to reflect the new topology, traffic is unable to use any alternate paths that may exist.

Normally, static routes are used only in simple network topologies. However, there are additional circumstances when static routing can be attractive. For example, static routes can be used:

To manually define a default route. This route is used to forward traffic when the routing table does not contain a more specific route to the destination.

To define a route that is not automatically advertised within a network.

When utilization or line tariffs make it undesirable to send routing advertisement traffic through lower-capacity WAN connections.

When complex routing policies are required. For example, static routes can be used to guarantee that traffic destined for a specific host traverses a designated network path.

To provide a more secure network environment. The administrator is aware of all subnetworks defined in the environment. The administrator specifically authorizes all communication permitted between these subnetworks.

To provide more efficient resource utilization. This method of routing table management requires no network bandwidth to advertise routes between neighboring devices. It also uses less processor memory and CPU cycles to calculate network paths.

#### 5.2.2 Distance vector routing

Distance vector algorithms are examples of dynamic routing protocols. These algorithms allow each device in the network to automatically build and maintain a local IP routing table.

The principle behind distance vector routing is simple. Each router in the internetwork maintains the *distance* or *cost* from itself to every known destination. This value represents the overall desirability of the path. Paths associated with a smaller cost value are more attractive to use than paths associated with a larger value. The path represented by the smallest cost becomes the preferred path to reach the destination.

This information is maintained in a *distance vector table.* The table is periodically

advertised to each neighboring router. Each router processes these advertisements to determine the best paths through the network.

The main advantage of distance vector algorithms is that they are typically easy to implement and debug. They are very useful in small networks with limited redundancy. However, there are several disadvantages with this type of protocol:

During an adverse condition, the length of time for every device in the network to produce an accurate routing table is called the *convergence time*. In large, complex internetworks using distance vector algorithms, this time can be excessive. While the routing tables are converging, networks are susceptible to inconsistent routing behavior. This can cause routing loops or other types of unstable packet forwarding.

To reduce convergence time, a limit is often placed on the maximum number of hops contained in a single route. Valid paths exceeding this limit are not usable in distance vector networks.

Distance vector routing tables are periodically transmitted to neighboring devices. They are sent even if no changes have been made to the contents of the table. This can cause noticeable periods of increased utilization in reduced capacity environments.

Enhancements to the basic distance vector algorithm have been developed to reduce the convergence and instability exposures. We describe these enhancements in 5.3.5, “Convergence and counting to infinity” on page 185.

RIP is a popular example of a distance vector routing protocol.

#### 5.2.3 Link state routing

The growth in the size and complexity of networks in recent years has necessitated the development of more robust routing algorithms. These algorithms address the shortcoming observed in distance vector protocols.

These algorithms use the principle of a *link state* to determine network topology. A link state is the description of an interface on a router (for example, IP address, subnet mask, type of network) and its relationship to neighboring routers. The collection of these link states forms a link state database.

The process used by link state algorithms to determine network topology is straightforward:

1. Each router identifies all other routing devices on the directly connected networks.
2. Each router advertises a list of all directly connected network links and the associated cost of each link. This is performed through the exchange of link state advertisements (LSAs) with other routers in the network.
3. Using these advertisements, each router creates a database detailing the current network topology. The topology database in each router is identical.
4. Each router uses the information in the topology database to compute the most desirable routes to each destination network. This information is used to update the IP routing table.

##### Shortest-Path First (SPF) algorithm

The SPF algorithm is used to process the information in the topology database. It provides a tree-representation of the network. The device running the SPF algorithm is the root of the tree. The output of the algorithm is the list of shortest-paths to each destination network. Figure 5-3 on page 178 provides an example of the shortest-path algorithm executed on router A.

*Figure 5-3 Shortest-Path First (SPF) example*

A

B

C

D

Link State

Database

4

2

1

D

1

3

3

E

A

B

C

D

B-2

C-1

A-2

D-4

A-1

D-1

E-3

C-1

B-4

E-3

C-3

D-3

A

B

C

D

E

Because each router is processing the same set of LSAs, each router creates an identical link state database. However, because each device occupies a different place in the network topology, the application of the SPF algorithm produces a different tree for each router.

The OSPF protocol is a popular example of a link state routing protocol.

#### 5.2.4 Path vector routing

Path vector routing is discussed in RFC 1322; the following paragraphs are based on the RFC.

The path vector routing algorithm is somewhat similar to the distance vector algorithm in the sense that each border router advertises the destinations it can reach to its neighboring router. However, instead of advertising networks in terms of a destination and the distance to that destination, networks are advertised as destination addresses and path descriptions to reach those destinations.

A route is defined as a pairing between a destination and the attributes of the path to that destination, thus the name, path vector routing, where the routers receive a vector that contains paths to a set of destinations.

The path, expressed in terms of the domains (or confederations) traversed so far, is carried in a special path attribute that records the sequence of routing domains through which the reachability information has passed. The path represented by the smallest number of domains becomes the preferred path to reach the destination.

The main advantage of a path vector protocol is its flexibility. There are several other advantages regarding using a path vector protocol:

The computational complexity is smaller than that of the link state protocol. The path vector computation consists of evaluating a newly arrived route and comparing it with the existing one, while conventional link state computation requires execution of an SPF algorithm.

Path vector routing does not require all routing domains to have homogeneous policies for route selection; route selection policies used by one routing domain are not necessarily known to other routing domains. The support for heterogeneous route selection policies has serious implications for the computational complexity. The path vector protocol allows each domain to make its route selection autonomously, based only on local policies. However, path vector routing can accommodate heterogeneous route selection with little additional cost.

Only the domains whose routes are affected by the changes have to recompute.

Suppression of routing loops is implemented through the path attribute, in contrast to link state and distance vector, which use a globally-defined monotonically thereby increasing metric for route selection. Therefore, different confederation definitions are accommodated because looping is avoided by the use of full path information.

Route computation precedes routing information dissemination. Therefore, only routing information associated with the routes selected by a domain is distributed to adjacent domains.

Path vector routing has the ability to selectively hide information.

However, there are disadvantages to this approach, including: Topology changes only result in the recomputation of routes affected by these

changes, which is more efficient than complete recomputation. However, because of the inclusion of full path information with each distance vector, the effect of a topology change can propagate farther than in traditional distance vector algorithms.

Unless the network topology is fully meshed or is able to appear so, routing loops can become an issue.

BGP is a popular example of a path vector routing protocol.

#### 5.2.5 Hybrid routing

The last category of routing protocols is hybrid protocols. These protocols attempt to combine the positive attributes of both distance vector and link state protocols. Like distance vector, hybrid protocols use metrics to assign a preference to a route. However, the metrics are more accurate than conventional distance vector protocols. Like link state algorithms, routing updates in hybrid protocols are event driven rather than periodic. Networks using hybrid protocols tend to converge more quickly than networks using distance vector protocols. Finally, these protocols potentially reduce the costs of link state updates and distance vector advertisements.

Although open hybrid protocols exist, this category is almost exclusively associated with the proprietary EIGRP algorithm. EIGRP was developed by Cisco Systems, Inc.

### 5.3 Routing Information Protocol (RIP)

RIP is an example of an interior gateway protocol designed for use within small autonomous systems. RIP is based on the Xerox XNS routing protocol. Early implementations of RIP were readily accepted because the code was incorporated in the Berkeley Software Distribution (BSD) UNIX-based operating system. RIP is a distance vector protocol.

In mid-1988, the IETF issued RFC 1058 with updates in RFC2453, which describes the standard operations of a RIP system. However, the RFC was issued after many RIP implementations had been completed. For this reason, some RIP systems do not support the entire set of enhancements to the basic distance vector algorithm (for example, poison reverse and triggered updates).

#### 5.3.1 RIP packet types

The RIP protocol specifies two packet types. These packets can be sent by any device running the RIP protocol:

Request packets: A request packet queries neighboring RIP devices to obtain their distance vector table. The request indicates if the neighbor should return either a specific subset or the entire contents of the table.

Response packets: A response packet is sent by a device to advertise the information maintained in its local distance vector table. The table is sent during the following situations:

* The table is automatically sent every 30 seconds.
* The table is sent as a response to a request packet generated by another RIP node.
* If triggered updates are supported, the table is sent when there is a change to the local distance vector table. We discuss triggered updates in “Triggered updates” on page 188.

When a response packet is received by a device, the information contained in the update is compared against the local distance vector table. If the update contains a lower cost route to a destination, the table is updated to reflect the new path.

#### 5.3.2 RIP packet format

RIP uses a specific packet format to share information about the distances to known network destinations. RIP packets are transmitted using UDP datagrams. RIP sends and receives datagrams using UDP port 520.

RIP datagrams have a maximum size of 512 octets. Updates larger than this size must be advertised in multiple datagrams. In LAN environments, RIP datagrams are sent using the MAC all-stations broadcast address and an IP network broadcast address. In point-to-point or non-broadcast environments, datagrams are specifically addressed to the destination device.

The RIP packet format is shown in Figure 5-4.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Octets  Request=1  1Response=2  Version = 1  1  2  Address Family  2Identifier for IP  2   |  |  | | --- | --- | | Command |  | | Version | | Reserved | | AFI: X'0002' | | Reserved | | IP Address | | Reserved | | Metric | |  |   4 }Routing Entry: May be repeated  }  8  4 |

*Figure 5-4 RIP packet format*

A 512 byte packet size allows a maximum of 25 routing entries to be included in a single RIP advertisement.

#### 5.3.3 RIP modes of operation

RIP hosts have two modes of operation:

Active mode: Devices operating in active mode advertise their distance vector table and also receive routing updates from neighboring RIP hosts. Routing devices are typically configured to operate in active mode.

Passive (or silent) mode: Devices operating in this mode simply receive routing updates from neighboring RIP devices. They do not advertise their distance vector table. End stations are typically configured to operate in passive mode.

#### 5.3.4 Calculating distance vectors

The distance vector table describes each destination network. The entries in this table contain the following information:

The destination network (vector) described by this entry in the table.

The associated cost (distance) of the most attractive path to reach this destination. This provides the ability to differentiate between multiple paths to a destination. In this context, the terms distance and cost can be misleading. They have no direct relationship to physical distance or monetary cost.

The IP address of the next-hop device used to reach the destination network.

Each time a routing table advertisement is received by a device, it is processed to determine if any destination can be reached by a lower cost path. This is done using the RIP distance vector algorithm. The algorithm can be summarized as:

At router initialization, each device contains a distance vector table listing each directly attached networks and configured cost. Typically, each network is assigned a cost of 1. This represents a single hop through the network. The total number of hops in a route is equal to the total cost of the route. However, cost can be changed to reflect other measurements such as utilization, speed, or reliability.

Each router periodically (typically every 30 seconds) transmits its distance vector table to each of its neighbors. The router can also transmit the table when a topology change occurs. Each router uses this information to update its local distance vector table:

* The total cost to each destination is calculated by adding the cost reported in a neighbor's distance vector table to the cost of the link to that neighbor. The path with the least cost is stored in the distance vector table.
* All updates automatically supersede the previous information in the distance vector table. This allows RIP to maintain the integrity of the routes in the routing table.

The IP routing table is updated to reflect the least-cost path to each destination.

Figure 5-5 illustrates the distance vector tables for three routers within a simple internetwork.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N2  R1 | | | | N3  N4  N5  N6  R2  R3  R4  R5 |
| N1  Router R2  Distance Vector  Table | | | | Router R3 Router R4  Distance Vector Distance Vector  Table Table   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Net | Next Hop | Metric |  | Net | Next  Hop | Metric | | N1 | R2 | 3 | N1 | R3 | 4 | | N2 | R2 | 2 | N2 | R3 | 3 | | N3 | Direct | 1 | N3 | R3 | 2 | | N4 | Direct | 1 | N4 | Direct | 1 | | N5 | R4 | 2 | N5 | Direct | 1 | | N6 | R4 | 3 | N6 | R5 | 2 | |
|  | Net | Next Hop | Metric |
|  | N1 | R1 | 2 |
|  | N2 | Direct | 1 |
|  | N3 | Direct | 1 |
|  | N4 | R3 | 2 |
|  | N5 | R3 | 3 |
|  | N6 | R3 | 4 |
|  | | | |

*Figure 5-5 A sample distance vector routing table*

#### 5.3.5 Convergence and counting to infinity

Given sufficient time, this algorithm will correctly calculate the distance vector table on each device. However, during this convergence time, erroneous routes may propagate through the network. Figure 5-6 shows this problem.

*Figure 5-6 Counting to infinity sample network*

Target

Network

A

B

C

D

(

n) = Network Cost

(1)

(1)

(1)

(1)

(1)

(10)

This network contains four interconnected routers. Each link has a cost of 1, except for the link connecting router C and router D; this link has a cost of 10. The costs have been defined so that forwarding packets on the link connecting router C and router D is undesirable. After the network has converged, each device has routing information describing all networks.

For example, to reach the target network, the routers have the following information:

Router D to the target network: Directly connected network. Metric is 1.

Router B to the target network: Next hop is router D. Metric is 2.

Router C to the target network: Next hop is router B. Metric is 3. Router A to the target network: Next hop is router B. Metric is 3.

Consider an adverse condition where the link connecting router B and router D fails. After the network has reconverged, all routes use the link connecting router C and router D to reach the target network. However, this reconvergence time can be considerable. Figure 5-7 illustrates how the routes to the target network are updated throughout the reconvergence period. For simplicity, this figure assumes all routers send updates at the same time.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time D: | Direct | 1 | Direct | 1 | Direct 1 | Direct | 1 | -- | Direct | 1 | Direct | 1 |
| B: | Unreachable |  | C | 4 | C 5 | C | 6 |  | C | 11 | C | 12 |
| C: | B | 3 | A | 4 | A 5 | A | 6 |  | A | 11 | D | 11 |
| A: | B | 3 | C | 4 | C 5 | C | 6 | -- | C | 11 | C | 12 |

*Figure 5-7 Network convergence sequence*

Reconvergence begins when router B notices that the route to router D is unavailable. Router B is able to immediately remove the failed route because the link has timed out. However, a considerable amount of time passes before the other routers remove their references to the failed route. This is described in the sequence of updates shown in Figure 5-7:

1. Prior to the adverse condition occurring, router A and router C have a route to the target network through router B.
2. The adverse condition occurs when the link connecting router D and router B fails. Router B recognizes that its preferred path to the target network is now invalid.
3. Router A and router C continue to send updates reflecting the route through router B. This route is actually invalid because the link connecting router D and router B has failed.
4. Router B receives the updates from router A and router C. Router B believes it should now route traffic to the target network through either router A or router C. In reality, this is not a valid route, because the routes in router A and router C are vestiges of the previous route through router B.
5. Using the routing advertisement sent by router B, router A and router C are able to determine that the route through router B has failed. However, router A and router C now believe the preferred route exists through the partner.

Network convergence continues as router A and router C engage in an extended period of mutual deception. Each device claims to be able to reach the target network through the partner device. The path to reach the target network now contains a routing loop.

The manner in which the costs in the distance vector table increment gives rise to the term *counting to infinity*. The costs continues to increment, theoretically to infinity. To minimize this exposure, whenever a network is unavailable, the incrementing of metrics through routing updates must be halted as soon as it is practical to do so. In a RIP environment, costs continue to increment until they reach a maximum value of 16. This limit is defined in RFC 1058.

A side effect of the metric limit is that it also limits the number of hops a packet can traverse from source network to destination network. In a RIP environment, any path exceeding 15 hops is considered invalid. The routing algorithm will discard these paths.

There are two enhancements to the basic distance vector algorithm that can minimize the counting to infinity problem:

Split horizon with poison reverse Triggered updates

These enhancements do not impact the maximum metric limit.

##### Split horizon

The excessive convergence time caused by counting to infinity can be reduced with the use of split horizon. This rule dictates that routing information is prevented from exiting the router on an interface through which the information was received.

The basic split horizon rule is not supported in RFC 1058. Instead, the standard specifies the enhanced split horizon with poison reverse algorithm. The basic rule is presented here for background and completeness. The enhanced algorithm is reviewed in the next section.

The incorporation of split horizon modifies the sequence of routing updates shown in Figure 5-7 on page 186. The new sequence is shown in Figure 5-8. The tables show that convergence occurs considerably faster using the split horizon rule.

Time

D:

Direct

1

Direct

1

Direct

1

Direct

1

B:

Unreachable

Unreachable

Unreachable

C

12

C:

B

3

A

4

D

11

D

11

A:

B

3

C

4

Unreachable

C

12

Note: Faster Routing Table Convergence

*Figure 5-8 Network convergence with split horizon*

The limitation to this rule is that each node must wait for the route to the unreachable destination to time out before the route is removed from the distance vector table. In RIP environments, this timeout is at least three minutes after the initial outage. During that time, the device continues to provide erroneous information to other nodes about the unreachable destination. This propagates routing loops and other routing anomalies.

##### Split horizon with poison reverse

Poison reverse is an enhancement to the standard split horizon implementation. It is supported in RFC 1058. With poison reverse, all known networks are advertised in each routing update. However, those networks learned through a specific interface are advertised as unreachable in the routing announcements sent out to that interface.

This drastically improves convergence time in complex, highly-redundant environments. With poison reverse, when a routing update indicates that a network is unreachable, routes are immediately removed from the routing table. This breaks erroneous, looping routes before they can propagate through the network. This approach differs from the basic split horizon rule where routes are eliminated through timeouts.

Poison reverse has no benefit in networks with no redundancy (single path networks).

One disadvantage to poison reverse is that it might significantly increase the size of routing annoucements exchanged between neighbors. This is because all routes in the distance vector table are included in each announcement. Although this is generally not an issue on local area networks, it can cause periods of increased utilization on lower-capacity WAN connections.

##### Triggered updates

Like split horizon with poison reverse, algorithms implementing triggered updates are designed to reduce network convergence time. With triggered updates, whenever a router changes the cost of a route, it immediately sends the modified distance vector table to neighboring devices. This mechanism ensures that topology change notifications are propagated quickly, rather than at the normal periodic interval.

Triggered updates are supported in RFC 1058.

#### 5.3.6 RIP limitations

There are a number of limitations observed in RIP environments:

Path cost limits: The resolution to the counting to infinity problem enforces a maximum cost for a network path. This places an upper limit on the maximum network diameter. Networks requiring paths greater than 15 hops must use an alternate routing protocol.

Network-intensive table updates: Periodic broadcasting of the distance vector table can result in increased utilization of network resources. This can be a concern in reduced-capacity segments.

Relatively slow convergence: RIP, like other distance vector protocols, is relatively slow to converge. The algorithms rely on timers to initiate routing table advertisements.

No support for variable length subnet masking: Route advertisements in a RIP environment do not include subnet masking information. This makes it impossible for RIP networks to deploy variable length subnet masks.

### 5.4 Routing Information Protocol Version 2 (RIP-2)

The IETF recognizes two versions of RIP:

RIP Version 1 (RIP-1): This protocol is described in RFC 1058.

RIP Version 2 (RIP-2): RIP-2 is also a distance vector protocol designed for use within an AS. It was developed to address the limitations observed in RIP-1. RIP-2 is described in RFC 2453. The standard (STD 56) was published in late 1994.

In practice, the term RIP refers to RIP-1. Whenever you encounter the term RIP in TCP/IP literature, it is safe to assume that the reference is to RIP Version 1 unless otherwise stated. This same convention is used in this document. However, when the two versions are being compared, the term RIP-1 is used to avoid confusion.

RIP-2 is similar to RIP-1. It was developed to extend RIP-1 functionality in small networks. RIP-2 provides these additional benefits not available in RIP-1:

Support for CIDR and VLSM: RIP-2 supports supernetting (that is, CIDR) and variable-length subnet masking. This support was the major reason the new standard was developed. This enhancement positions the standard to accommodate a degree of addressing complexity not supported in RIP-1.

Support for multicasting: RIP-2 supports the use of multicasting rather than simple broadcasting of routing annoucements. This reduces the processing load on hosts not listening for RIP-2 messages. To ensure interoperability with RIP-1 environments, this option is configured on each network interface.

Support for authentication: RIP-2 supports authentication of any node transmitting route advertisements. This prevents fraudulent sources from corrupting the routing table.

Support for RIP-1: RIP-2 is fully interoperable with RIP-1. This provides backward-compatibility between the two standards.

As noted in the RIP-1 section, one notable shortcoming in the RIP-1 standard is the implementation of the metric field. RIP-1 specifies the metric as a value between 0 and 16. To ensure compatibility with RIP-1 networks, RIP-2 preserves this definition. In both standards, networks paths with a hop-count greater than 15 are interpreted as unreachable.

#### 5.4.1 RIP-2 packet format

The original RIP-1 specification was designed to support future enhancements. The RIP-2 standard was able to capitalize on this feature. RIP-2 developers noted that a RIP-1 packet already contains a version field and that 50% of the octets are unused.

Figure 5-9 illustrates the contents of a RIP-2 packet. The packet is shown with authentication information. The first entry in the update contains either a routing entry or an authentication entry. If the first entry is an authentication entry, 24 additional routing entries can be included in the message. If there is no authentication information, 25 routing entries can be provided.

*Figure 5-9 RIP-2 packet format*

Command

Version

Reserved

AFI: X'FFFF'

Authentication Type

Authentication Data

AFI:2

Route Tag

IP Address

Subnet Mask

Next Hop

Metric

Number of Octets

Request=1

Response=2

0=

No Authentication

2=

Password Data

Password if Type 2 Selected

Routing Entry: May not be

repeated

}

}

1

1

2

2

2

16

2

2

4

4

4

4

}

}

Authentication

Entry

The use of the command field, IP address field, and metric field in a RIP-2 message is identical to the use in a RIP-1 message. Otherwise, the changes implemented in a RIP-2 packets include:

**Version** The value contained in this field must be two. This instructs RIP-1 routers to ignore any information contained in the previously unused fields.

**AFI (Address Family)** A value of x’0002’ indicates the address contained in the network address field is an IP address. An value of x'FFFF' indicates an authentication entry.

**Authentication Type** This field defines the remaining 16 bytes of the

authentication entry. A value of 0 indicates *no* authentication. A value of two indicates the authentication data field contains password data.

**Authentication Data** This field contains a 16-byte password.

**Route Tag** This field is intended to differentiate between internal and external routes. Internal routes are learned through RIP-2 within the same network or AS.

**Subnet Mask** This field contains the subnet mask of the referenced network.

**Next Hop** This field contains a recommendation about the next hop the router should use when sending datagrams to the referenced network.

#### 5.4.2 RIP-2 limitations

RIP-2 was developed to address many of the limitations observed in RIP-1. However, the path cost limits and slow convergence inherent in RIP-1 networks are also concerns in RIP-2 environments.

In addition to these concerns, there are limitations to the RIP-2 authentication process. The RIP-2 standard does not encrypt the authentication password. It is transmitted in clear text. This makes the network vulnerable to attack by anyone with direct physical access to the environment.

### 5.5 RIPng for IPv6

RIPng was developed to allow routers within an IPv6-based network to exchange information used to compute routes. It is documented in RFC 2080. We provide additional information regarding IPv6 in 9.1, “IPv6 introduction” on page 328.

Like the other protocols in the RIP family, RIPng is a distance vector protocol designed for use within a small autonomous system. RIPng uses the same algorithms, timers, and logic used in RIP-2.

RIPng has many of the same limitations inherent in other distance vector

protocols. Path cost restrictions and convergence time remain a concern in

RIPng networks.

#### 5.5.1 Differences between RIPng and RIP-2

There are two important distinctions between RIP-2 and RIPng:

Support for authentication: The RIP-2 standard includes support for authenticating a node transmitting routing information. RIPng does not include any native authentication support. Rather, RIPng uses the security features inherent in IPv6. In addition to authentication, these security features provide the ability to encrypt each RIPng packet. This can control the set of devices that receive the routing information.One consequence of using IPv6 security features is that the AFI field within the RIPng packet is eliminated. There is no longer a need to distinguish between authentication entries and routing entries within an advertisement.

Support for IPv6 addressing formats: The fields contained in RIPng packets were updated to support the longer IPv6 address format.

#### 5.5.2 RIPng packet format

RIPng packets are transmitted using UDP datagrams. RIPng sends and receives datagrams using UDP port number 521.

The format of a RIPng packet is similar to the RIP-2 format. Specifically, both packets contain a 4 octet command header followed by a set of 20 octet route entries. The RIPng packet format is shown in Figure 5-10.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Octets  Request=1   |  |  | | --- | --- | | Command |  | | Version | | Reserved | | Route Table Entry (RTE) | |  |   1 {Response=2  1  2  20 {May be repeated |

*Figure 5-10 RIPng packet format*

The use of the command field and the version field is identical to the use in a RIP-2 packet. However, the fields containing routing information have been updated to accommodate the 16 octet IPv6 address. These fields are used differently than the corresponding fields in a RIP-1 or RIP-2 packet. The format of the RTE is shown in Figure 5-11.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Octets   |  |  | | --- | --- | | IPv6 Prefix |  | | Route Tag | | Prefix Length | | Metric | |  |   16  2  1  1 |

*Figure 5-11 Route table entry (RTE)*

In RIPng, the combination of the IP prefix and the prefix length identifies the route to be advertised. The metric remains encoded in a 1 octet field. This length is sufficient because RIPng uses a maximum hop-count of 16.

Another difference between RIPng and RIP-2 is the process used to determine the next hop. In RIP-2, each route table entry contains a next hop field. In RIPng, including this information in each RTE would have doubled the size of the advertisement. Therefore, in RIPng, the next hop is included in a special type of RTE. The specified next hop applies to each subsequent routing table entry in the advertisement. The format of an RTE used to specify the next hop is shown in Figure 5-12.

16

2

1

1

IPv6 Next Hop Address

Reserved

Reserved

Metric 0x'FF'

Number of Octets

{

Used to distinguish a

next hop entry

*Figure 5-12 Next Hop route table entry (RTE)*

The next hop RTE is identified by a value of 0x’FF’ in the metric field. This reserved value is outside the valid range of metrics.

The use of RTEs and next hop RTEs is shown in Figure 5-13.

*Figure 5-13 Using the RIPng RTE*

4

20

20

20

20

20

20

20

20

Number of Octets

Command

Routing entry #1

Routing entry #2

Routing entry #3

Next hop RTE A

Routing entry #4

Routing entry #5

Next hop RTE B

Routing entry #6

In this example, the first three routing entries do not have a corresponding next hop RTE. The address prefixes specified by these entries will be routed through the advertising router. The prefixes included in routing entries 4 and 5 will route through the next hop address specified in the next hop RTE A. The prefix included in routing entry 6 will route through the next hop address specified in the next hop RTE B.

### 5.6 Open Shortest Path First (OSPF)

The Open Shortest Path First (OSPF) protocol is another example of an interior gateway protocol. It was developed as a non-proprietary routing alternative to address the limitations of RIP. Initial development started in 1988 and was finalized in 1991. Subsequent updates to the protocol continue to be published. The current version of the standard is documented in RFC 2328.

OSPF provides a number of features not found in distance vector protocols. Support for these features has made OSPF a widely-deployed routing protocol in large networking environments. In fact, RFC 1812 – Requirements for IPv4 Routers, lists OSPF as the only required dynamic routing protocol. The following features contribute to the continued acceptance of the OSPF standard:

Equal cost load balancing: The simultaneous use of multiple paths can provide more efficient utilization of network resources.

Logical partitioning of the network: This reduces the propagation of outage information during adverse conditions. It also provides the ability to aggregate routing announcements that limit the advertisement of unnecessary subnet information.

Support for authentication: OSPF supports the authentication of any node transmitting route advertisements. This prevents fraudulent sources from corrupting the routing tables.

Faster convergence time: OSPF provides instantaneous propagation of routing changes. This expedites the convergence time required to update network topologies.

Support for CIDR and VLSM: This allows the network administrator to efficiently allocate IP address resources.

OSPF is a link state protocol. As with other link state protocols, each OSPF router executes the SPF algorithm (“Shortest-Path First (SPF) algorithm” on page 177) to process the information stored in the link state database. The algorithm produces a shortest-path tree detailing the preferred routes to each destination network.

#### 5.6.1 OSPF terminology

OSPF uses specific terminology to describe the operation of the protocol.

##### OSPF areas

OSPF networks are divided into a collection of *areas*. An area consists of a logical grouping of networks and routers. The area can coincide with geographic or administrative boundaries. Each area is assigned a 32-bit *area ID*.

Subdividing the network provides the following benefits:

Within an area, every router maintains an identical topology database describing the routing devices and links within the area. These routers have no knowledge of topologies outside the area. They are only aware of routes to these external destinations. This reduces the size of the topology database maintained by each router.

Areas limit the potentially explosive growth in the number of link state updates. Most LSAs are distributed only within an area.

Areas reduce the CPU processing required to maintain the topology database. The SPF algorithm is limited to managing changes within the area.

###### Backbone area and area 0

All OSPF networks contain at least one area. This area is known as area 0 or the backbone area. Additional areas can be created based on network topology or other design requirements.

In networks containing multiple areas, the backbone physically connects to all other areas. OSPF expects all areas to announce routing information directly into the backbone. The backbone then announces this information into other areas.

Figure 5-14 on page 198 depicts a network with a backbone area and four additional areas.

##### Intra-area, area border, and AS boundary routers

There are three classifications of routers in an OSPF network. Figure 5-14 illustrates the interaction of these devices.

Area 4

Area 2

Area 1

ASBR

ABR

ABR

ABR

Area 3

ABR

ASBR

IA

IA

AS External Links

Area 0

AS 10

AS External Links

Key

ASBR - AS Border Router

ABR - Area Border Router

IA - Intra-Area Router

*Figure 5-14 OSPF router types*

Where:

|  |  |
| --- | --- |
| **Intra-area routers** | This class of router is logically located entirely within an OSPF area. Intra-area routers maintain a topology database for their local area. |
| **Area border routers (ABR)** | This class of router is logically connected to two or more areas. One area must be the backbone area. An ABR is used to interconnect areas. They maintain a separate topology database for each attached area. ABRs also execute  separate instances of the SPF algorithm for each area. |

**AS boundary routers (ASBR)** This class of router is located at the periphery of an OSPF internetwork. It functions as a gateway exchanging reachability between the OSPF network and other routing environments.

ASBRs are responsible for announcing AS external link advertisements through the AS. We provide more information about external link advertisements in 5.6.4, “OSPF route redistribution” on page 208.

Each router is assigned a 32-bit *router ID* (RID). The RID uniquely identifies the device. One popular implementation assigns the RID from the lowest-numbered IP address configured on the router.

##### Physical network types

OSPF categorizes network segments into three types. The frequency and types of communication occurring between OSPF devices connected to these networks is impacted by the network type:

Point-to-point: Point-to-point networks directly link two routers.

Multi-access: Multi-access networks support the attachment of more than two routers.

They are further subdivided into two types:

* Broadcast networks have the capability of simultaneously directing a packet to all attached routers. This capability uses an address that is recognized by all devices. Ethernet and token-ring LANs are examples of OSPF broadcast multi-access networks.
* Non-broadcast networks do not have broadcasting capabilities. Each packet must be specifically addressed to every router in the network. X.25 and frame relay networks are examples of OSPF non-broadcast multi-access networks.

Point-to-multipoint: Point-to-multipoint networks are a special case of multi-access, non-broadcast networks. In a point-to-multipoint network, a device is not required to have a direct connection to every other device. This is known as a partially meshed environment.

##### Neighbor routers and adjacencies

Routers that share a common network segment establish a neighbor relationship

on the segment. Routers must agree on the following information to become neighbors:

Area ID: The routers must belong to the same OSPF area.

Authentication: If authentication is defined, the routers must specify the same password.

Hello and dead intervals: The routers must specify the same timer intervals used in the Hello protocol. We describe this protocol further in “OSPF packet types” on page 203.

Stub area flag: The routers must agree that the area is configured as a stub area. We describe stub areas further in 5.6.5, “OSPF stub areas” on page 210.

After two routers have become neighbors, an adjacency relationship can be formed between the devices. Neighboring routers are considered adjacent when they have synchronized their topology databases. This occurs through the exchange of link state information.

##### Designated and backup designated router

The exchange of link state information between neighbors can create significant quantities of network traffic. To reduce the total bandwidth required to synchronize databases and advertise link state information, a router does not necessarily develop adjacencies with every neighboring device:

Multi-access networks: Adjacencies are formed between an individual router and the (backup) designated router.

Point-to-point networks: An adjacency is formed between both devices.

Each multi-access network elects a designated router (DR) and backup designated router (BDR). The DR performs two key functions on the network segment:

It forms adjacencies with all routers on the multi-access network. This causes the DR to become the focal point for forwarding LSAs.

It generates network link advertisements listing each router connected to the multi-access network. For additional information regarding network link advertisements, see “Link state advertisements and flooding” on page 201.

The BDR forms the same adjacencies as the designated router. It assumes DR functionality when the DR fails.

Each router is assigned an 8-bit priority, indicating its ability to be selected as the DR or BDR. A router priority of zero indicates that the router is not eligible to be selected. The priority is configured on each interface in the router.

Figure 5-15 illustrates the relationship between neighbors. No adjacencies are formed between routers that are not selected to be the DR or BDR.

DR

Other

Other

BDR

Adjacent

Neighbors

Neighbors

*Figure 5-15 Relationship between adjacencies and neighbors*

##### Link state database

The link state database is also called the *topology database*. It contains the set of link state advertisements describing the OSPF network and any external connections. Each router within the area maintains an identical copy of the link state database.

**Note:** RFC 2328 uses the term link state database in preference to topology database. The former term has the advantage in that it describes the contents of the database. The latter term is more descriptive of the purpose of the database. This book has previously used the term topology database for this reason. However for the remainder of the OSPF section, we refer to it as the link state database.

##### Link state advertisements and flooding

The contents of an LSA describe an individual network component (that is, router, segment, or external destination). LSAs are exchanged between adjacent OSPF routers. This is done to synchronize the link state database on each device.

When a router generates or modifies an LSA, it must communicate this change throughout the network. The router starts this process by forwarding the LSA to each adjacent device. Upon receipt of the LSA, these neighbors store the information in their link state database and communicate the LSA to their neighbors. This store and forward activity continues until all devices receive the update. This process is called *reliable flooding*. Two steps are taken to ensure that this flooding effectively transmits changes without overloading the network with excessive quantities of LSA traffic:

Each router stores the LSA for a period of time before propagating the information to its neighbors. If, during that time, a new copy of the LSA arrives, the router replaces the stored version. However, if the new copy is outdated, it is discarded.

To ensure reliability, each link state advertisement must be acknowledged. Multiple acknowledgements can be grouped together into a single acknowledgement packet. If an acknowledgement is not received, the original link state update packet is retransmitted.

Link state advertisements contain five types of information. Together these advertisements provide the necessary information needed to describe the entire OSPF network and any external environments:

Router LSAs: This type of advertisement describes the state of the router's interfaces (links) within the area. They are generated by every OSPF router. The advertisements are flooded throughout the area.

Network LSAs: This type of advertisement lists the routers connected to a multi-access network. They are generated by the DR on a multi-access segment. The advertisements are flooded throughout the area.

Summary LSAs (Type-3 and Type-4): This type of advertisement is generated by an ABR. There are two types of summary link advertisements:

* Type-3 summary LSAs describe routes to destinations in other areas within the OSPF network (inter-area destinations).
* Type-4 summary LSAs describe routes to ASBRs. Summary LSAs are used to exchange reachability information between areas. Normally, information is announced into the backbone area. The backbone then injects this information into other areas.

AS external LSAs: This type of advertisement describes routes to destinations external to the OSPF network. They are generated by an ASBR. The advertisements are flooded throughout all areas in the OSPF network.

Figure 5-16 illustrates the different types of link state advertisements.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Router Links | Network Links | | Router  - Advertised by router - Describes state/cost of router's links | DR  - Advertised by designated router - Describes all routers attached to network | | Summary Links  ABR  Area X  Area 0   * Advertised by router * Describes state/cost of | External Links  ASBR  Area 0  Area X   * Advertised by router * Describes state/cost of |   router's links router's links |

*Figure 5-16 OSPF link state advertisements*

##### OSPF packet types

OSPF packets are transmitted in IP datagrams. They are not encapsulated within TCP or UDP packets. The IP header uses protocol identifier 89. OSPF packets are sent with an IP ToS of 0 and an IP precedence of internetwork control. This is used to obtain preferential processing for the packets. We discuss ToS and IP precedence further in “Integrated Services” on page 288.

Wherever possible, OSPF uses multicast facilities to communicate with neighboring devices. In broadcast and point-to-point environments, packets are sent to the reserved multicast address 224.0.0.5. RFC 2328 refers to this as the AllSPFRouters address. In non-broadcast environments, packets are addressed to the neighbor’s specific IP address.

All OSPF packets share the common header shown in Figure 5-17. The header provides general information including area identifier, RID, checksum, and authentication information.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Octets  Version = 2  1  1= Hello   |  |  | | --- | --- | | Version |  | | Packet Type | | Packet Length | | Router ID | | Area ID | | Checksum | | Authentication Type | | Authentication Data | |  |   1 {2=Database Description  3=Link State Request  4=Link State Update  25=Link State Acknowledgement  4  4  2  20=No Authentication  {1=Simple Password  8Password if Type 1 Selected |

*Figure 5-17 OSPF common header*

The type field identifies the OSPF packet as one of five possible types:

|  |  |
| --- | --- |
| **Hello** | This packet type discovers and maintains neighbor relationships. |
| **Database description** | This packet type describes the set of LSAs contained in the router's link state database. |
| **Link state request** | This packet type requests a more current instance of an LSA from a neighbor. |
| **Link state update** | This packet type provides a more current instance of an LSA to a neighbor. |

###### Link state acknowledgement

This packet type acknowledges receipt of a newly received LSA.

We describe the use of these packets in the next section.

#### 5.6.2 Neighbor communication

OSPF is responsible for determining the optimum set of paths through a network. To accomplish this, each router exchanges LSAs with other routers in the network. The OSPF protocol defines a number of activities to accomplish this information exchange:

Discovering neighbors

Electing a designated router

Establishing adjacencies and synchronizing databases

The five OSPF packet types are used to support these information exchanges.

##### Discovering neighbors: The OSPF Hello protocol

The Hello protocol discovers and maintains relationships with neighboring routers. Hello packets are periodically sent out to each router interface. The packet contains the RID of other routers whose hello packets have already been received over the interface.

When a device sees its own RID in the hello packet generated by another router, these devices establish a neighbor relationship.

The hello packet also contains the router priority, DR identifier, and BDR identifier. These parameters are used to elect the DR on multi-access networks.

##### Electing a designated router

All multi-access networks must have a DR. A BDR can also be selected. The backup ensures there is no extended loss of routing capability if the DR fails.

The DR and BDR are selected using information contained in hello packets. The device with the highest OSPF router priority on a segment becomes the DR for that segment. The same process is repeated to select the BDR. In case of a tie, the router with the highest RID is selected. A router declared the DR is ineligible to become the BDR.

After elected, the DR and BDR proceed to establish adjacencies with all routers on the multi-access segment.

##### Establishing adjacencies and synchronizing databases

Neighboring routers are considered adjacent when they have synchronized their link state databases. A router does not develop an adjacency with every neighboring device. On multi-access networks, adjacencies are formed only with the DR and BDR. This is a two step process.

###### Step 1: Database exchange process

The first phase of database synchronization is the database exchange process.

This occurs immediately after two neighbors attempt to establish an adjacency. The process consists of an exchange of database description packets. The packets contain a list of the LSAs stored in the local database.

During the database exchange process, the routers form a master/subordinate relationship. The master is the first to transmit. Each packet is identified by a sequence number. Using this sequence number, the subordinate acknowledges each database description packet from the master. The subordinate also includes its own set of link state headers in the acknowledgements.

###### Step 2: Database loading

During the database exchange process, each router notes the link state headers for which the neighbor has a more current instance (all advertisements are time stamped). After the process is complete, each router requests the more current information from the neighbor. This request is made with a link state request packet.

When a router receives a link state request, it must reply with a set of link state update packets providing the requested LSA. Each transmitted LSA is acknowledged by the receiver. This process is similar to the reliable flooding procedure used to transmit topology changes throughout the network.

Every LSA contains an age field indicating the time in seconds since the origin of the advertisement. The age continues to increase after the LSA is installed in the topology database. It also increases during each hop of the flooding process. When the maximum age is reached, the LSA is no longer used to determining routing information and is discarded from the link state database. This age is also used to distinguish between two otherwise identical copies of an advertisement.

#### 5.6.3 OSPF neighbor state machine

The OSPF specification defines a set of neighbor states and the events that can cause a neighbor to transition from one state to another. A state machine is used to describe these transitions:

Down: This is the initial state. It indicates that no recent information has been received from any device on the segment.

Attempt: This state is used on non-broadcast networks. It indicates that a neighbor appears to be inactive. Attempts continue to reestablish contact.

Init: Communication with the neighbor has started, but bidirectional communication has not been established. Specifically, a hello packet was received from the neighbor, but the local router was not listed in the neighbor's hello packet.

2-way: Bidirectional communication between the two routers has been established. Adjacencies can be formed. Neighbors are eligible to be elected as designated routers.

ExStart: The neighbors are starting to form an adjacency.

Exchange: The two neighbors are exchanging their topology databases.

Loading: The two neighbors are synchronizing their topology databases.

Full: The two neighbors are fully adjacent and their databases are synchronized.

Network events cause a neighbor’s OSPF state to change. For example, when a router receives a hello packet from a neighboring device, the OSPF neighbor state changes from Down to Init. When bidirectional communication has been established, the neighbor state changes from Init to 2-Way. RFC 2328 contains a complete description of the events causing a state change.

##### OSPF virtual links and transit areas

Virtual links are used when a network does not support the standard OSPF network topology. This topology defines a backbone area that directly connects to each additional OSPF area. The virtual link addresses two conditions:

It can logically connect the backbone area when it is not contiguous.

It can connect an area to the backbone when a direct connection does not exist.

A virtual link is established between two ABRs sharing a common non-backbone area. The link is treated as a point-to-point link. The common area is known as a *transit area*. Figure 5-18 on page 208 illustrates the interaction between virtual links and transit areas when used to connect an area to the backbone.

*Figure 5-18 OSPF virtual link and transit areas*

Area 0

Area 1

Area 2

Transit Area

ABR

ABR

Virtual link

This diagram shows that area 1 does not have a direct connection to the backbone. Area 2 can be used as a transit area to provide this connection. A virtual link is established between the two ABRs located in area 2. Establishing this virtual link logically extends the backbone area to connect to area 1.

A virtual link is used only to transmit routing information. It does not carry regular traffic between the remote area and the backbone. This traffic, in addition to the virtual link traffic, is routed using the standard intra-area routing within the transit area.

#### 5.6.4 OSPF route redistribution

Route redistribution is the process of introducing external routes into an OSPF network. These routes can be either static routes or routes learned through another routing protocol. They are advertised into the OSPF network by an ASBR. These routes become OSPF external routes. The ASBR advertises these routes by flooding OSPF AS external LSAs throughout the entire OSPF network.

The routes describe an end-to-end path consisting of two portions:

External portion: This is the portion of the path external to the OSPF network. When these routes are distributed into OSPF, the ASBR assigns an initial cost. This cost represents the *external cost* associated with traversing the external portion of the path.

Internal portion: This is the portion of the path internal to the OSPF network. Costs for this portion of the network are calculated using standard OSPF algorithms.

OSPF differentiates between two types of external routes. They differ in the way the cost of the route is calculated. The ASBR is configured to redistribute the route as:

External type 1: The total cost of the route is the sum of the external cost and any internal OSPF costs.

External type 2: The total cost of the route is always the external cost. This ignores any internal OSPF costs required to reach the ASBR.

Figure 5-19 illustrates an example of the types of OSPF external routes.

*Figure 5-19 OSPF route redistribution*

**R1 Routing Table**

10.99.5.0/24

E1: Cost 60

or

E2: Cost 50

**R2 Routing Table**

10.99.5.0/24

E1: Cost 65

or

E2: Cost 50

R1

R2

(20)

(10)

(15)

OSPF

Network

ASBR

10.99.5.0/24

redistributed

with external cost 50

External

Internal

RIP

Network

10.99.5.0/24

In this example, the ASBR is redistributing the 10.99.5.0/24 route into the OSPF network. This subnet is located within the RIP network. The route is announced into OSPF with an external cost of 50. This represents the cost for the portion of the path traversing the RIP network:

If the ASBR redistributed the route as an E1 route, R1 will contain an external route to this subnet with a cost of 60 (50 + 10). R2 will have an external route with a cost of 65 (50 + 15).

If the ASBR redistributed the route as an E2 route, both R1 and R2 will contain an external route to this subnet with a cost of 50. Any costs associated with traversing segments within the OSPF network are not included in the total cost to reach the destination.

#### 5.6.5 OSPF stub areas

OSPF allows certain areas to be defined as a stub area. A stub area is created when the ABR connecting to a stub area excludes AS external LSAs from being flooded into the area. This is done to reduce the size of the link state database maintained within the stub area routers. Because there are no specific routes to external networks, routing to these destinations is based on a default route generated by the ABR. The link state databases maintained within the stub area contain only the default route and the routes from within the OSPF environment (for example, intra-area and inter-area routes).

Because a stub area does not allow external LSAs, a stub area cannot contain an ASBR. No external routes can be generated from within the stub area.

Stub areas can be deployed when there is a single exit point connecting the area to the backbone. An area with multiple exit points can also be a stub area. However, there is no guarantee that packets exiting the area will follow an optimal path. This is due to the fact that each ABR generates a default route. There is no ability to associate traffic with a specific default routes.

All routers within the area must be configured as stub routers. This configuration is verified through the exchange of hello packets.

##### Not-so-stubby areas

An extension to the stub area concept is the *not-so-stubby area* (NSSA). This alternative is documented in RFC 3101. An NSSA is similar to a stub area in that the ABR servicing the NSSA does not flood any external routes into the NSSA. The only routes flooded into the NSSA are the default route and any other routes from within the OSPF environment (for example, intra-area and inter-area).

However, unlike a stub area, an ASBR can be located within an NSSA. This

ASBR can generate external routes. Therefore, the link state databases

maintained within the NSSA contain the default route, routes from within the OSPF environment (for example, intra-area and inter-area routes), and the external routes generated by the ASBR within the area.

The ABR servicing the NSSA floods the external routes from within the NSSA throughout the rest of the OSPF network.

#### 5.6.6 OSPF route summarization

Route summarization is the process of consolidating multiple contiguous routing entries into a single advertisement. This reduces the size of the link state database and the IP routing table. In an OSPF network, summarization is performed at a border router. There are two types of summarization:

Inter-area route summarization: Inter-area summarization is performed by the ABR for an area. It is used to summarize route advertisements originating within the area. The summarized route is announcement into the backbone. The backbone receives the aggregated route and announces the summary into other areas.

External route summarization: This type of summarization applies specifically to external routes injected into OSPF. This is performed by the ASBR distributing the routes into the OSPF network. Figure 5-20 illustrates an example of OSPF route summarization.

*Figure 5-20 OSPF route summarization*

**OSPF**

**Area 1**

ASBR

10.99.192.0/24

through

10.99.254.0/24

RIP

Network

10.99.0.0/24

through

10.99.83.0/24

ABR

R1

**OSPF Area 2**

10.99.0.0/28

10.99.192.0/28

**OSPF**

**Area 0**

External Summary

10.99.0.0/28

Inter-area Summary

10.99.192.0/28

In this figure, the ASBR is advertising a single summary route for the 64 subnetworks located in the RIP environment. This single summary route is flooded throughout the entire OSPF network. In addition, the ABR is generating a single summary route for the 64 subnetworks located in area 1. This summary route is flooded through area 0 and area 2. Depending of the configuration of the ASBR, the inter-area summary route can also be redistributed into the RIP network.

### 5.7 Enhanced Interior Gateway Routing Protocol (EIGRP)

The Enhanced Interior Gateway Routing Protocol (EIGRP) is categorized as a hybrid routing protocol. Similar to a distance vector algorithm, EIGRP uses metrics to determine network paths. However, like a link state protocol, topology updates in an EIGRP environment are event driven.

EIGRP, as the name implies, is an interior gateway protocol designed for use within an AS. In properly designed networks, EIGRP has the potential for improved scalability and faster convergence over standard distance vector algorithms. EIGRP is also better positioned to support complex, highly redundant networks.

EIGRP is a proprietary protocol developed by Cisco Systems, Inc. At the time of this writing, it is not an IETF standard protocol.

#### 5.7.1 Features of EIGRP

EIGRP has several capabilities. Some of these capabilities are also available in distance vector or link state algorithms.

EIGRP maintains a list of alternate routes that can be used if a preferred path fails. When the path fails, the new route is immediately installed in the IP routing table. No route recomputation is performed.

EIGRP allows partial routing updates. When EIGRP discovers a neighboring router, each device exchanges their entire routing table. After the initial information exchange, only routing table changes are propagated. There is no periodic rebroadcasting of the entire routing table.

EIGRP uses a low amount of bandwidth. During normal network operations, only hello packets are transmitted through a stable network.

EIGRP supports supernetting (CIDR) and variable length subnet masks (VLSM). This enables the network administrator to efficiently allocate IP address resources.

EIGRP supports the ability to summarize routing annoucements. This limits the advertisement of unnecessary subnet information.

EIGRP can provide network layer routing for multiple protocols such as AppleTalk, IPX, and IP networks.

EIGRP supports the simultaneous use of multiple unequal cost paths to a destination. Each route is installed in the IP routing table. EIGRP also intelligently load balances traffic over the multiple paths.

EIGRP uses a topology table to install routes into the IP routing table. The topology table lists all destination networks currently advertised by neighboring routers. The table contains all the information needed to build a set of distances and vectors to each destination.

EIGRP maintains a table to track the state of each adjacent neighbor. This is called a neighbor table.

EIGRP can guarantee the ordered delivery of packets to a neighbor. However, not all types of packets must be reliably transmitted. For example, in a network that supports multicasting, there is no need to send individual, acknowledged hello packets to each neighbor. To provide efficient operation, reliability is provided only when needed. This improves convergence time in networks containing varying speed connections.

##### Neighbor discovery and recovery

EIGRP can dynamically learn about other routers on directly attached networks. This is similar to the Hello protocol used for neighbor discovery in an OSPF environment.

Devices in an EIGRP network exchange hello packets to verify each neighbor is operational. Like OSPF, the frequency used to exchange packets is based on the network type. Packets are exchanged at a five second interval on high bandwidth links (for example, LAN segments). Otherwise, hello packets on lower bandwidth connections are exchanged every 60 seconds.

Also like OSPF, EIGRP uses a hold timer to remove inactive neighbors. This timer indicates the amount of time that a device will continue to consider a neighbor active without receiving a hello packet from the neighbor.

##### EIGRP routing algorithm

EIGRP does not rely on periodic updates to converge on the topology. Instead, it builds a topology table containing each of its neighbor’s advertisements. Unlike a distance vector protocol, this data is not discarded.

EIGRP processes the information in the topology table to determine the best paths to each destination network. EIGRP implements an algorithm known as Diffusing Update ALgorithm (DUAL).

##### Route recomputation

For a specific destination, the successor is the neighbor router currently used for packet forwarding. This device has the least-cost path to the destination and is guaranteed not to be participating in a routing loop. A feasible successor assumes forwarding responsibility when the current successor router fails. The set of feasible successors represent the devices that can become a successor without requiring a route recomputation or introducing routing loops.

A route recomputation occurs when there is no known feasible successor to the destination. The successor is the neighbor router currently used for packet forwarding. The process starts with a router sending a multicast query packet to determine if any neighbor is aware of a feasible successor to the destination. A neighbor replies if it has an feasible successor.

If the neighbor does not have a feasible successor, the neighbor can return a query indicating it also is performing a route recomputation. When the link to a neighbor fails, all routes that used that neighbor as the only feasible successor require a route recomputation.

#### 5.7.2 EIGRP packet types

EIGRP uses five types of packets to establish neighbor relationships and advertise routing information:

Hello/acknowledgement: These packets are used for neighbor discovery. They are multicast advertised on each network segment. Unicast responses to the hello packet are returned. A hello packet without any data is considered an acknowledgement.

Updates: These packets are used to convey reachability information for each destination. When a new neighbor is discovered, unicast update packets are exchanged to allow each neighbor to build their topology table. Other types of advertisements (for example, metric changes) use multicast packets. Update packets are always transmitted reliably.

Queries and replies: These packets are exchanged when a destination enters an active state. A multicast query packet is sent to determine if any neighbor contains a feasible successor to the destination. Unicast reply packets are sent to indicate that the neighbor does not need to go into an active state because a feasible successor has been identified. Query and reply packets are transmitted reliably.

Request: These packets are used to obtain specific information from a neighbor. These packets are used in route server applications.

### 5.8 Exterior Gateway Protocol (EGP)

EGP is an exterior gateway protocol of historical merit. It was one of the first protocols developed for communication between autonomous systems. It is described in RFC 904.

EGP assumes the network contains a single backbone and a single path exists between any two autonomous systems. Due to this limitation, the current use of EGP is minimal. In practice, EGP has been replaced by BGP.

EGP is based on periodic polling using a hello/I-hear-you message exchange.

These are used to monitor neighbor reachability and solicit update responses.

The gateway connecting to an AS is permitted to advertise only those destination networks reachable within the local AS. It does not advertise reachability information about its EGP neighbors outside the AS.

### 5.9 Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) is an exterior gateway protocol. It was originally developed to provide a loop-free method of exchanging routing information between autonomous systems. BGP has since evolved to support aggregation and summarization of routing information.

BGP is an IETF draft standard protocol described in RFC 4271. The version described in this RFC is BGP Version 4. Following standard convention, this document uses the term BGP when referencing BGP Version 4.

#### 5.9.1 BGP concepts and terminology

BGP uses specific terminology to describe the operation of the protocol. Figure 5-21 illustrates this terminology.

AS1

AS2

AS3

ASX

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

OSPF/RIP

BGP Speaker

OSPF/RIP

BGP Speaker

OSPF/RIP

BGP Speaker

OSPF/RIP

BGP Speaker

OSPF/RIP

BGP Speaker

IBGP

IBGP

EBGP

EBGP

EBGP

*Figure 5-21 Components of a BGP network*

BGP uses the following terms:

BGP speaker: A router configured to support BGP.

BGP neighbors (peers): A pair of BGP speakers that exchange routing information. There are two types of BGP neighbors:

* Internal (IBGP) neighbor: A pair of BGP speakers within the same AS.
* External (EBGP) neighbor: A pair of BGP neighbors, each in a different

AS. These neighbors typically share a directly connected network.

BGP session: A TCP session connecting two BGP neighbors. The session is used to exchange routing information. The neighbors monitor the state of the

session by sending keepalive messages.1

Traffic type: BGP defines two types of traffic:

* Local: Traffic local to an AS either originates or terminates within the AS. Either the source or the destination IP address resides in the AS.
* Transit: Any traffic that is not local traffic is transit traffic. One of the goals of BGP is to minimize the amount of transit traffic.

AS type: BGP defines three types of autonomous systems:

* Stub: A stub AS has a single connection to one other AS. A stub AS carries only local traffic.
* Multihomed: A multihomed AS has connections to two or more autonomous systems. However, a multihomed AS has been configured so that it does not forward transit traffic.
* Transit: A transit AS has connections to two or more autonomous systems and carries both local and transit traffic. The AS can impose policy restrictions on the types of transit traffic that will be forwarded.

Depending on the configuration of the BGP devices within AS 2 in Figure 5-21 on page 216, this autonomous system can be either a multihomed AS or a transit AS.

AS number: A 16-bit number uniquely identifying an AS.

AS path: A list of AS numbers describing a route through the network. A BGP neighbor communicates paths to its peers.

Routing policy: A set of rules constraining the flow of data packets through the network. Routing policies are not defined in the BGP protocol. Rather, they are used to configure a BGP device. For example, a BGP device can be configured so that:

* A multihomed AS can refuse to act as a transit AS. This is accomplished by advertising only those networks contained within the AS.
* A multihomed AS can perform transit AS routing for a restricted set of adjacent autonomous systems. It does this by tailoring the routing advertisements sent to EBGP peers.
* An AS can optimize traffic to use a specific AS path for certain categories of traffic.

Network layer reachability information (NLRI): NLRI is used by BGP to advertise routes. It consists of a set of networks represented by the tuple

<length,prefix>. For example, the tuple <14,220.24.106.0> represents the CIDR route 220.24.106.0/14.

1 This keepalive message is implemented in the application layer. It is independent of the keepalive message available in many TCP implementations.

Routes and paths: A route associates a destination with a collection of attributes describing the path to the destination. The destination is specified in NRLI format. The path is reported as a collection of path attributes. This information is advertised in UPDATE messages. For additional information describing the UPDATE message, see 5.9.3, “Protocol description” on page 220.

#### 5.9.2 IBGP and EBGP communication

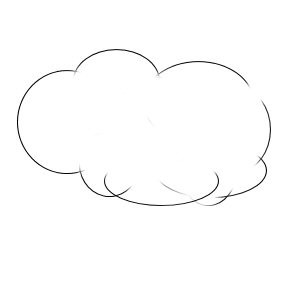
BGP does not replace the IGP operating within an AS. Instead, it cooperates with the IGP to establish communication between autonomous systems. BGP within an AS is used to advertise the local IGP routes. These routes are advertised to BGP peers in other autonomous systems. Figure 5-22 on page 219 illustrates the communication that occurs between BGP peers. This example shows four autonomous systems. AS 2, AS 3, and AS 4 each have an EBGP connection to AS 1. A full mesh of IBGP sessions exists between BGP devices within AS 1.

Network 10.0.0.0/8 is located within AS 3. Using BGP, the existence of this network is advertised to the rest of the environment:

R4 in AS 3 uses its EBGP connection to announce the network to AS 1.

R1 in AS 1 uses its IBGP connections to announce the network to R2 and R3.

R2 in AS 1 uses its EBGP session to announce the network into AS 2. R3 in AS 1 uses its EBGP session 5 to announce the network into AS 4.



BGP

R6

**AS 2**

BGP

R5

**AS 4**

BGP

R4

10.0.0.0/8

**AS 3**

BGP

R2

**AS 1**

BGP

R1

BGP

R3

IGP

Interconnection

IBGP

IBGP

IBGP

EBGP

EBGP

EBGP

*Figure 5-22 EBGP and IBGP communication*

Several additional operational issues are shown in Figure 5-22:

Role of BGP and the IGP: The diagram shows that while BGP alone carries information between autonomous systems, both BGP and the IGP are used to carry information through an AS.

Establishing the TCP session between peers: Before establishing a BGP session, a device verifies that routing information is available to reach the peer:

* EBGP peers: EBGP peers typically share a directly connected network. The routing information needed to exchange BGP packets between these peers is trivial.
* IBGP peers: IBGP peers can be located anywhere within the AS. They do not need to be directly connected. BGP relies on the IGP to locate a peer.

Packet forwarding between IBGP peers uses IGP-learned routes.

Full mesh of BGP sessions within an AS: IBGP speakers assume a full mesh of BGP sessions have been established between peers in the same AS. In Figure 5-22 on page 219, all three BGP peers in AS 1 are interconnected with BGP sessions.

When a BGP speaker receives a route update from an IBGP peer, the receiving speaker uses EBGP to propagate the update to external peers. Because the receiving speaker assumes a full mesh of IBGP sessions have been established, it does not propagate the update to other IBGP peers.

For example, assume that there was no IBGP session between R1 and R3 in Figure\_82. R1 receives the update about 10.0.0.0/8 from AS 3. R1 forwards the update to its BGP peers, namely R2. R2 receives the IBGP update and forwards it to its EBGP peers, namely R6. No update is sent to R3. If R3 needs to receive this information, R1 and R3 must be configured to be BGP peers.

#### 5.9.3 Protocol description

BGP establishes a reliable TCP connection between peers. Sessions are established using TCP port 179. BGP assumes the transport connection will manage fragmentation, retransmission, acknowledgement, and sequencing.

When two speakers initially form a BGP session, they exchange their entire routing table. This routing information contains the complete AS path used to reach each destination. The information avoids the routing loops and counting-to-infinity behavior observed in RIP networks. After the entire table has been exchanged, changes to the table are communicated as incremental updates.

##### BGP packet types

All BGP packets contain a standard header. The header specifies the BGP packet type. The valid BGP packet types include:

OPEN[[2]](#footnote-2): This message type establishes a BGP session between two peer nodes.

UPDATE: This message type transfers routing information between GP peers.

NOTIFICATION: This message is sent when an error condition is detected.

KEEPALIVE: This message determines if peers are reachable.

Figure 5-23 shows the flow of these message types between two autonomous systems.

BGP

BGP

ASY

BGP

ASX

Open

Keep Alive

Update

Notification

*Figure 5-23 BGP message flows between BGP speakers*

##### Opening and confirming a BGP connection

After a TCP session has been established between two peer nodes, each router sends an OPEN message to the neighbor. The OPEN message includes:

The originating router's AS number and BGP router identifier.

A suggested value for the hold timer. We discuss the function of this timer in the next section.

Optional parameters. This information is used to authenticate a peer.

An OPEN message contains support for authenticating the identity of a BGP peer. However, the BGP standard does not specify a specific authorization mechanism. This allows BGP peers to select any supported authorization scheme.

An OPEN message is acknowledged by a KEEPALIVE message. After peer routers have established a BGP connection, they can exchange additional information.

##### Maintaining the BGP connection

BGP does not use any transport-based keepalive to determine if peers are reachable. Instead, BGP messages are periodically exchanged between peers. If no messages are received from the peer for the duration specified by the hold timer, the originating router assumes that an error has occurred. When this happens, an error notification is sent to the peer and the connection is closed.

RFC 4271 recommends a 90 second hold timer and a 30 second keepalive timer.

##### Sending reachability information

Reachability information is exchanged between peers in UPDATE messages.

BGP does not require a periodic refresh of the entire BGP routing table. Therefore, each BGP speaker must retain a copy of the current BGP routing table used by each peer. This information is maintained for the duration of the connection. After neighbors have performed the initial exchange of complete routing information, only incremental updates to that information are exchanged.

An UPDATE message is used to advertise feasible routes or withdraw infeasible routes. The message can simultaneously advertise a feasible route and withdraw multiple infeasible routes from service. Figure 5-24 depicts the format of an UPDATE message:

Network layer reachability information (NLRI).

Path attributes (we discuss path attributes in “Path attributes” on page 223.

Withdrawn routes.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Octets   |  |  | | --- | --- | | Common Header |  | | Unfeasible Routes Length | | Withdrawn Routes | | Total Path Attribute Length | | Path Attributes | | Network Layer Reachability Information | |  |   19Type = 2  2  Variable  2  Variable  Variable |

*Figure 5-24 BGP UPDATE message*

Several path attributes can be used to describe a route.

###### Withdrawn routes

The unfeasible routes length field indicates the total length of the withdrawn routes field.

The withdrawn routes field provides a list of IP addresses prefixes that are not feasible or are no longer in service. These addresses need to be withdrawn from the BGP routing table. The withdrawn routes are represented in the same tuple-format as the NLRI.

##### Notification of error conditions

A BGP device can observe error conditions impacting the connection to a peer. NOTIFICATION messages are sent to the neighbor when these conditions are detected. After the message is sent, the BGP transport connection is closed. This means that all resources for the BGP connection are deallocated. The routing table entries associated with the remote peer are marked as invalid. Finally, other peers are notified that these routes are invalid.

Notification messages include an error code and an error subcode.The error codes provided by BGP include:

Message header error

OPEN message error

UPDATE message error

Hold timer expired

Finite state machine error

Cease

The error subcode further qualifies the specific error. Each error code can have multiple subcodes associated with it.

#### 5.9.4 Path selection

BGP is a path vector protocol. In path vector routing, the path is expressed in terms of the domains (or confederations) traversed so far. The best path is obtained by comparing the number of domains of each feasible route. However, inter-AS routing complicates this process. There are no universally agreed-upon metrics that can be used to evaluate external paths. Each AS has its own set of criteria for path evaluation.

##### Path attributes

Path attributes are used to describe and evaluate a route. Peers exchange path attributes along with other routing information. When a device advertises a route,

it can add or modify the path attributes before advertising the route to a peer. The combination of attributes are used to select the best path.

Each path attribute is placed into one of four separate categories:

Well-known mandatory: The attribute must be recognized by all BGP implementations. It must be sent in every UPDATE message.

Well-known discretionary: The attribute must be recognized by all BGP implementations. However, it is not required to be sent in every UPDATE message.

Optional transitive: It is not required that every BGP implementation recognize this type of attribute. A path with an unrecognized optional transitive attribute is accepted and simply forwarded to other BGP peers.

Optional non-transitive: It is not required that every BGP implementation recognize this type of attribute. These attributes can be ignored and not passed along to other BGP peers.

BGP defines seven attribute types to define an advertised route:

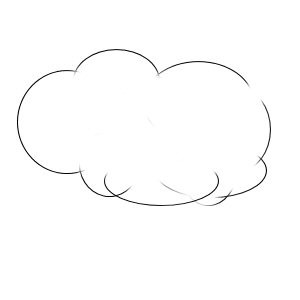
ORIGIN: This attribute defines the origin of the path information. Valid selections are IGP (interior to the AS), EGP, or INCOMPLETE. This is a well-known mandatory attribute.

AS\_PATH: This attribute defines the set of autonomous systems that must be traversed to reach the advertised network. Each BGP device prepends its AS number onto the AS path sequence before sending the routing information to an EBGP peer. Using the sample network depicted in Figure 5-22 on page 219, R4 advertises network 10.0.0.0 with an AS\_PATH of 3. When the update traverses AS 1, R2 prepends its own AS number to it. When the routing update reaches R6, the AS\_PATH attribute for network 10.0.0.0 is <1 3>. This is a well-known mandatory attribute.

NEXT\_HOP: This attribute defines the IP address of the next hop used to reach the destination. This is a well-known mandatory attribute.

For routing updates received over EBGP connections, the next hop is typically the IP address of the EBGP neighbor in the remote AS. BGP specifies that this next hop is passed without modification to each IBGP neighbor. As a result, each IBGP neighbor must have a route to reach the neighbor in the remote AS. Figure 5-25 on page 225 illustrates this interaction.

*Figure 5-25 NEXT\_HOP attribute*



AS 1

AS 3

BGP

R1

BGP

R3

BGP

R4

172.16.1.2

172.16.1.1

10.0.0.0/8

EBGP

IBGP

In this example, when a routing update for network 10.0.0.0/8 is sent from AS 3, R1 receives the update with the NEXT\_HOP attribute set to 172.16.1.1.

When this update is forwarded to R3, the next hop address remains 172.16.1.1. R3 must have appropriate routing information to reach this address. Otherwise, R3 will drop packets destined for AS 3 if the next hop is inaccessible.

MULTI\_EXIT\_DISC (multi-exit discriminator, MED): This attribute is used to discriminate among multiple exit points to a neighboring AS. If this information is received from an EBGP peer, it is propagated to each IBGP peer. This attribute is not propagated to peers in other autonomous systems. If all other attributes are equal, the exit point with the lowest MED value is preferred. This is an optional non-transitive attribute. MED is discussed further in RFC 4451.

LOCAL\_PREF (local preference): This attribute is used by a BGP speaker to inform other speakers within the AS of the originating speaker's degree of preference for the advertised route. Unlike MED, this attribute is used only within an AS. The value of the local preference is not distributed outside an AS. If all other attributes are equal, the route with the higher degree of preference is preferred. This is a well-known discretionary attribute.

ATOMIC\_AGGREGATE: This attribute is used when a BGP peer receives advertisements for the same destination identified in multiple, non-matching routes (that is, overlapping routes). One route describes a smaller set of destinations (a more specific prefix), other routes describe a larger set of destinations (a less specific prefix). This attribute is used by the BGP speaker to inform peers that it has selected the less specific route without selecting the more specific route. This is a well-known discretionary attribute. A route with this attribute included may actually traverse autonomous systems not listed in the AS\_PATH.

AGGREGATOR: This attribute indicates the last AS number that formed the aggregate route, followed by the IP address of the BGP speaker that formed the aggregate route. For further information about route aggregation, refer to

5.9.6, “BGP aggregation” on page 228. This is an optional transitive attribute.

##### Decision process

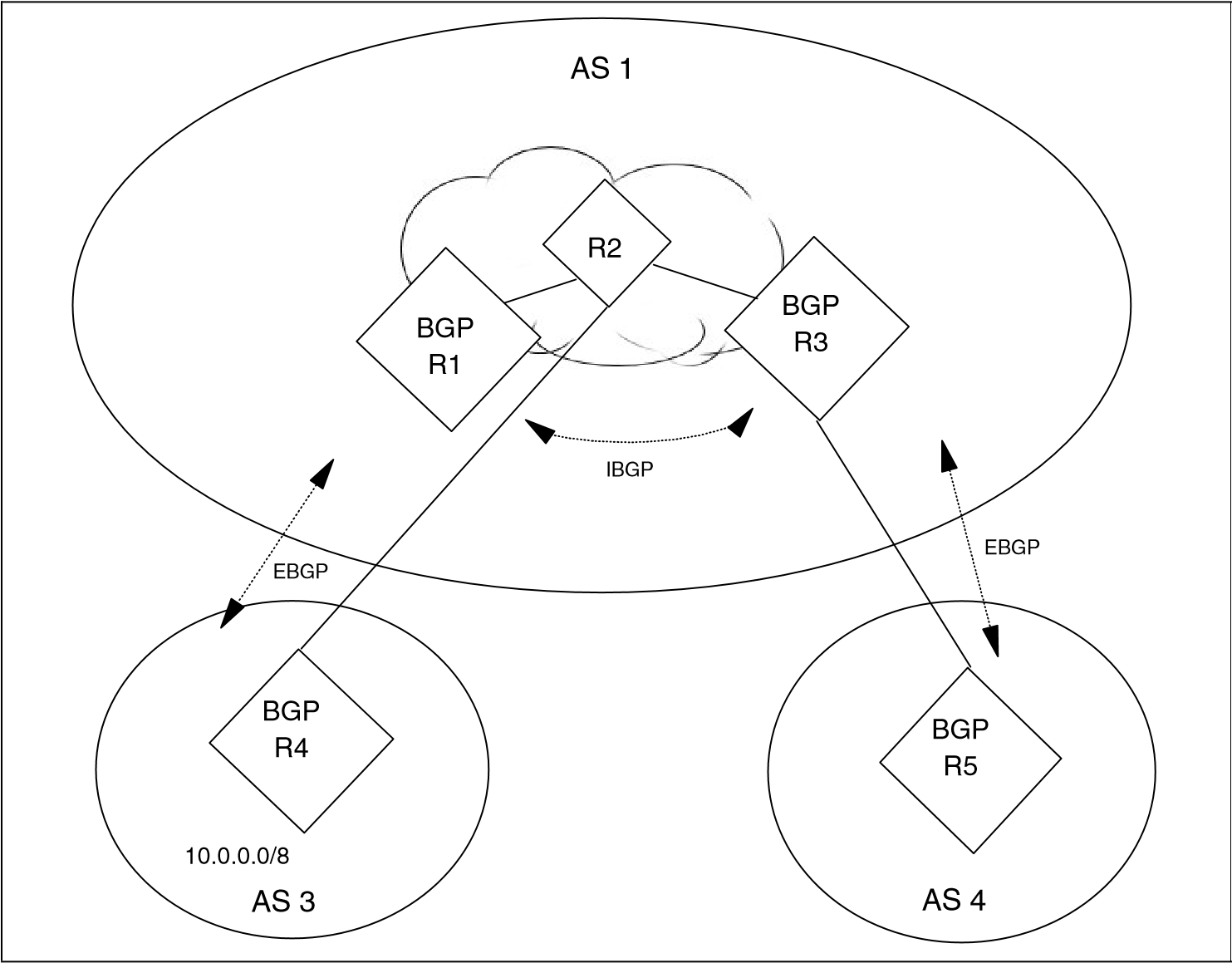
The process to select the best path uses the path attributes describing each route. The attributes are analyzed and a *degree of preference* is assigned. Because there can be multiple paths to a given destination, the route selection process determines the degree of preference for each feasible route. The path with the highest degree of preference is selected as the best path. This is the path advertised to each BGP neighbor. Route aggregation can also be performed during this process. Where there are multiple paths to a destination, BGP tracks each individual path. This allows faster convergence to the alternate path when the primary path fails.

#### 5.9.5 BGP synchronization

Figure 5-26 on page 227 shows an example of an AS providing transit service. In this example, AS 1 is used to transport traffic between AS 3 and AS 4. Within AS 1, R2 is not configured for BGP. However, R2 is used for communication between R1 and R3. Traffic between these two BGP nodes physically traverses through R2.

Using the routing update flow described earlier, the 10.0.0.0/8 network is advertised using the EBGP connection between R4 and R1. R1 passes the network advertisement to R3 using its existing IBGP connection. Because R2 is not configured for BGP, it is unaware of any networks in AS 3. A problem occurs if R3 needs to communicate with a device in AS 3. R3 passes the traffic to R2. However, because R2 does not have any routes to AS 3 networks, the traffic is dropped.

If R3 advertises the 10.0.0.0/8 network to AS 4, the problem continues. If AS 4 needs to communicate with a device in AS 3, the packets are forwarded from R5 to R3. R3 forwards the packets to R2 where they are discarded.



*Figure 5-26 BGP synchronization*

This situation is addressed by the synchronization rule of BGP. The rule states that a transit AS will not advertise a route before all routers within the AS have learned about the route. In this example, R3 will not advertise the existence of the networks in AS 3 until R2 has built a proper routing table.

There are three methods to implement the synchronization rule:

Enable BGP on all devices within the transit AS. In this solution, R2 has an IBGP session with both R1 and R3. R2 learns of the 10.0.0.0/8 network at the same time it is advertised to R3. At that time, R3 announces the routes to its peer in AS 4.

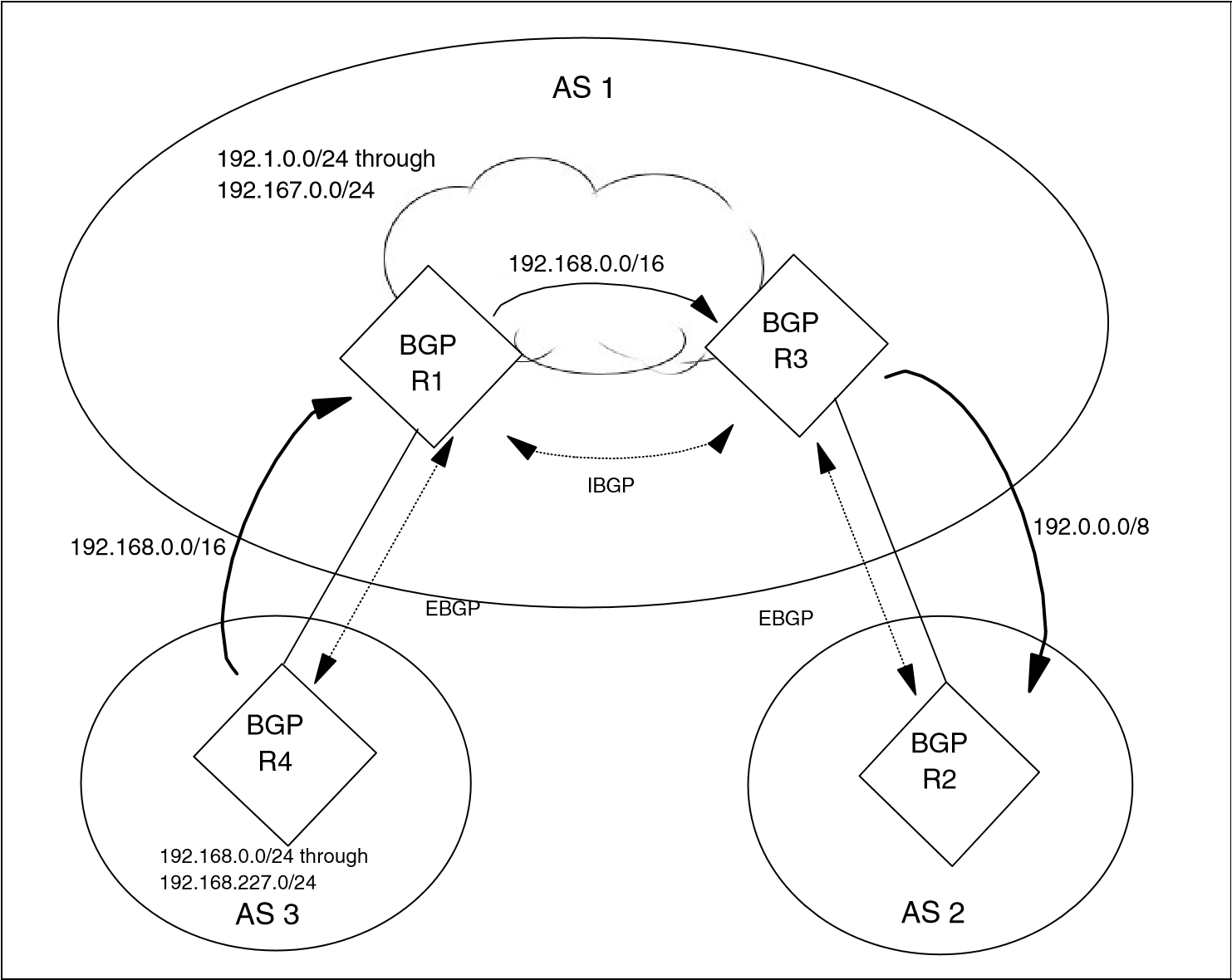
Redistribute the routes into the IGP used within the transit area. In this solution, R1 redistributes the 10.0.0.0/8 network into the IGP within AS 1. R3 learns of the network through two routing protocols: BGP and the IGP. After

R3 learns of the network through the IGP, it is certain that other routers within the AS have also learned of the routes. At that time, R3 announces the routes to its peer in AS 4.

Encapsulate the transit traffic across the AS. In this solution, transit traffic is encapsulated within IP datagrams addressed to the exit gateway. Because this does not require the IGP to carry exterior routing information, no synchronization is required between BGP and the IGP. R3 can immediately announce the routes to its peer in AS 4.

#### 5.9.6 BGP aggregation

The major improvement introduced in BGP Version 4 was support for CIDR and route aggregation. These features allow BGP peers to consolidate multiple contiguous routing entries into a single advertisement. It significantly enhances the scalability of BGP into large internetworking environments. Figure 5-27 on page 229 illustrates these functions.



*Figure 5-27 BGP route aggregation*

This diagrams depicts three autonomous systems interconnected by BGP. In this example, networks 192.168.0.0 through 182.168.227.0 are located within AS 3. To reduce the size of routing announcements, R4 aggregates these individual networks into a single route entry prior to advertising into AS 1. The single entry 192.168.0.0/16 represents a valid CIDR supernet even though it is an illegal Class C network.

BGP aggregate routes contain additional information within the AS\_PATH path attribute. When aggregate entries are generated from a set of more specific routes, the AS\_PATH attributes of the more specific routes are combined. For example, in Figure 5-27, the aggregate route 192.0.0.0/8 is announced from AS 1 into AS 2. This aggregate represents the set of more specific routes deployed within AS 1 and AS 3. When this aggregate route is sent to AS 2, the

AS\_PATH attribute consists of <1 3>. This is done to prevent routing information

loops. A loop can occur if AS 1 generated an aggregate with an AS\_PATH attribute of <1>. If AS 2 had a direct connection to AS 3, the route with the less-specific AS\_PATH advertised from AS 1 can generate a loop. This is because AS 2 does not know this aggregate contains networks located within

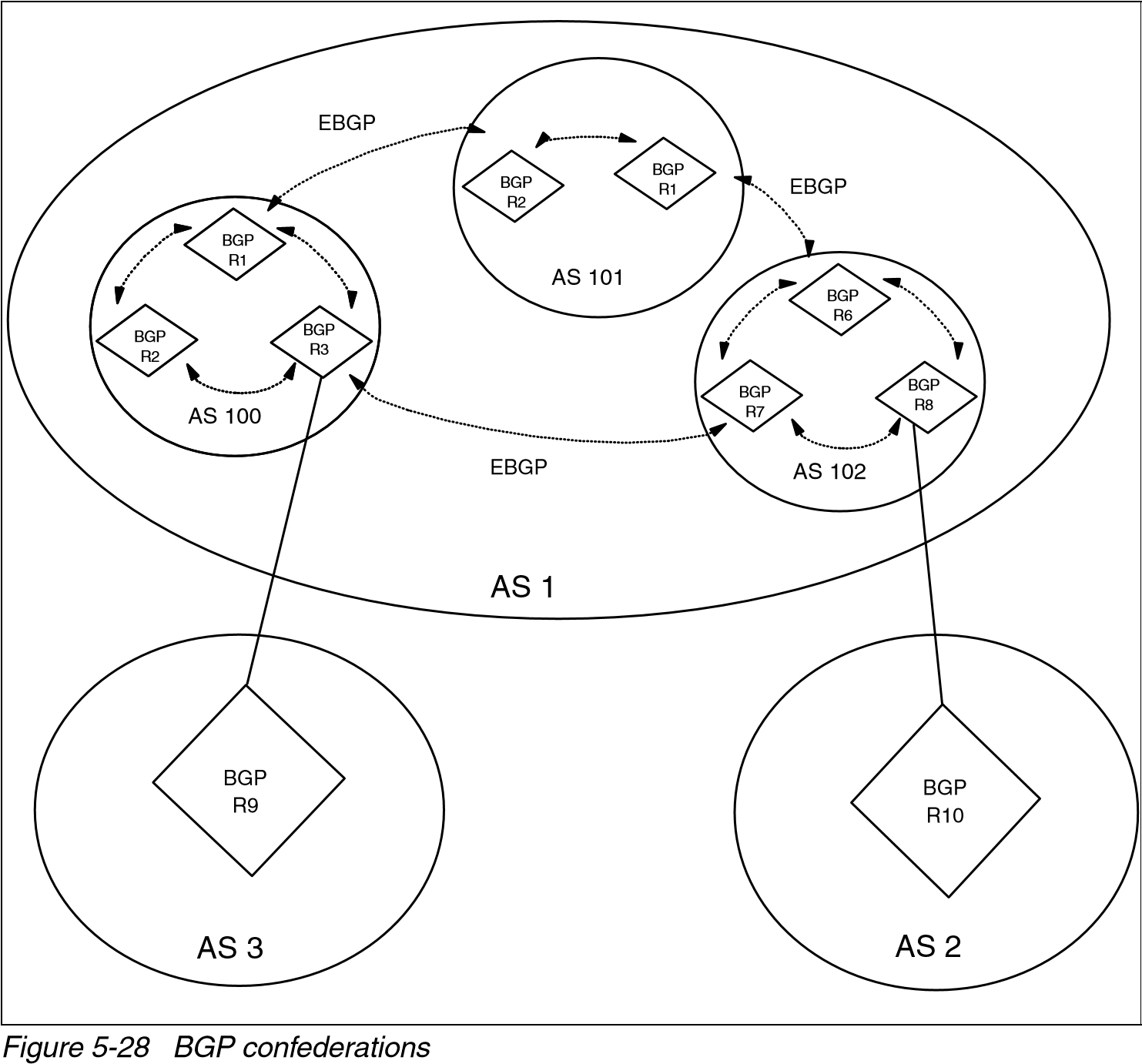
AS 3.

#### 5.9.7 BGP confederations

BGP requires that all speakers within a single AS have a fully meshed set of IBGP connections. This can be a scaling problem in networks containing a large number of IBGP peers. The use of BGP confederations addresses this problem.

A BGP confederation creates a set of autonomous systems that represent a single AS to peers external to the confederation. This removes the full mesh requirement and reduces management complexity.

Figure 5-28 illustrates the operation of a BGP confederation. In this sample network, AS 1 contains eight BGP speakers. A standard BGP network would require 28 IBGP sessions to fully mesh the speakers.



A confederation divides the AS into a set of domains. In this example, AS 1 contains three domains. Devices within a domain have a fully meshed set of IBGP connections. Each domain also has an EBGP connection to other domains within the confederation. In the example network, R1, R2, and R3 have fully meshed IBGP sessions. R1 has an EBGP session within the confederation to R4. R3 has an EBGP session outside the confederation to R9.

Each router in the confederation is assigned a confederation ID. A member of the confederation uses this ID in all communications with devices outside the confederation. In this example, each router is assigned a confederation ID of AS 1.

All communications from AS 1 to AS 2 or AS 3 appear to have originated from the confederation ID of AS 1. Even though communication between domains within a confederation occurs with EBGP, the domains exchange routing updates as though they were connected by IBGP. Specifically, the information contained in the NEXT\_HOP, MULTI\_EXIT\_DESC, and LOCAL\_PREF attributes is preserved between domains. The confederation appears to be a single AS to other autonomous systems.

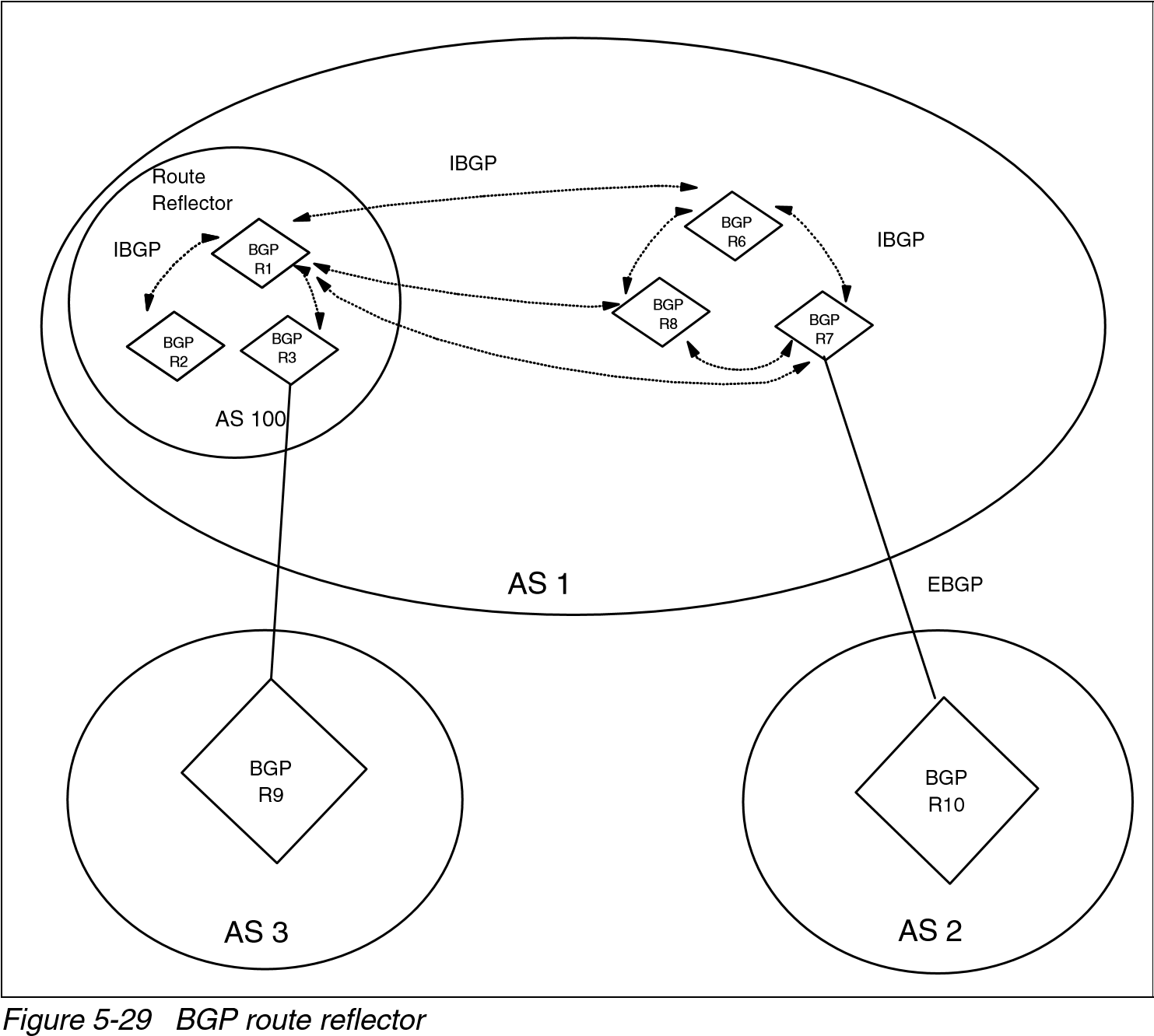
BGP confederations are described in RFC 3065. At the time of this writing, this is a proposed standard. Regardless, BGP confederations have been widely deployed throughout the Internet. Numerous vendors support this feature.

#### 5.9.8 BGP route reflectors

Route reflectors are another solution to address the requirement for a full mesh of IBGP sessions between peers in an AS. As noted previously, when a BGP speaker receives an update from an IBGP peer, the receiving speaker propagates the update only to EBGP peers. The receiving speaker does not forward the update to other IBGP peers.

Route reflectors relax this restriction. BGP speakers are permitted to advertise IBGP learned routes to certain IBGP peers. Figure 5-29 on page 232 depicts an environment using route reflectors. R1 is configured as a route reflector for R2 and R3. R2 and R3 are route reflector clients of R1. No IBGP session is defined between R2 and R3. When R3 receives an EBGP update from AS 3, it is passed to R1 using IBGP. Because R1 is configured as a reflector, R1 forwards the IBGP update to R2.

Figure 5-29 also illustrates the interaction between route reflectors and conventional BGP speakers within an AS.



In Figure 5-29, R1, R2, and R3 are in the route reflector domain. R6, R7, and R8 are conventional BGP speakers containing a full mesh of IBGP peer connections. In addition, each of these speakers is peered with the route reflector. This configuration permits full IBGP communication within AS 1.

Although not shown in Figure 5-29, an AS can contain more than one route reflector. When this occurs, each reflector treats other reflectors as a conventional IBGP peer.

Route reflectors are described inRFC 4456. At the time of this writing, this is a proposed standard.

### 5.10 Routing protocol selection

The choice of a routing protocol is a major decision for the network administrator.

It has a major impact on overall network performance. The selection depends on network complexity, size, and administrative policies.

The protocol chosen for one type of network might not be appropriate for other types of networks. Each unique environment must be evaluated against a number of fundamental design requirements:

Scalability to large environments: The potential growth of the network dictates the importance of this requirement. If support is needed for large, highly-redundant networks, consider link state or hybrid algorithms. Distance vector algorithms do not scale into these environments.

Stability during outages: Distance vector algorithms might introduce network instability during outage periods. The counting to infinity problems (5.3.5, “Convergence and counting to infinity” on page 185) can cause routing loops or other non-optimal routing paths. Link state or hybrid algorithms reduce the potential for these problems.

Speed of convergence: Triggered updates provide the ability to immediately initiate convergence when a failure is detected. All three types of protocols support this feature. One contributing factor to convergence is the time required to detect a failure. In OSPF and EIGRP networks, a series of hello packets must be missed before convergence begins. In RIP environments, subsequent route advertisements must be missed before convergence in initiated. These detection times increase the time required to restore communication.

Metrics: Metrics provide the ability to groom appropriate routing paths through the network. Link state algorithms consider bandwidth when calculating routes. EIGRP improves this to include network delay in the route calculation.

Support for VLSM: The availability of IP address ranges dictates the importance of this requirement. In environments with an constrained supply of addresses, the network administrator must develop an addressing scheme that intelligently overlays the network. VLSM is a major component of this plan. The use of private addresses ranges can also address this concern.

Vendor interoperability: The types of devices deployed in a network indicate the importance of this requirement. If the network contains equipment from a number of vendors, use standard routing protocols. The IETF has dictated the operating policies for the distance vector and link state algorithms described in this document. Implementing these algorithms avoids any interoperability problems encountered with nonstandard protocols.

Ease of implementation: Distance vector protocols are the simplest routing protocol to configure and maintain. Because of this, these protocols have the largest implementation base. Limited training is required to perform problem resolution in these environments.

In small, non-changing environments, static routes are also simple to implement. These definitions change only when sites are added or removed from the network. The administrator must assess the importance of each of these requirements when determining the appropriate routing protocol for an environment.

### 5.11 Additional functions performed by the router

The main functions performed by a router relate to managing the IP routing table and forwarding data. However, the router should be able to provide information alerting other devices to potential network problems.

This information is provided by the ICMP protocol described in 3.2, “Internet Control Message Protocol (ICMP)” on page 109. The information includes:

ICMP Destination Unreachable: The destination address specified in the IP packet references an unknown IP network.

ICMP Redirect: Redirect forwarding of traffic to a more suitable router along the path to the destination.

ICMP Source Quench: Congestion problems (for example, too many incoming datagrams for the available buffer space) have been encountered in a device along the path to the destination.

ICMP Time Exceeded: The Time-to-Live field of an IP datagram has reached zero. The packet is not able to be delivered to the final destination.

In addition, each IP router should support the following base ICMP operations and messages:

Parameter problem: This message is returned to the packet’s source if a problem with the IP header is found. The message indicates the type and location of the problem. The router discards the errored packet.

Address mask request/reply: A router must implement support for receiving ICMP Address Mask Request messages and responding with ICMP Address Mask Reply messages.

Timestamp: The router must return a Timestamp Reply to every Timestamp message that is received. It should be designed for minimum variability in delay. To synchronize the clock on the router, the UDP Time Server Protocol or the Network Time Protocol (NTP) can be used.

Echo request/reply: A router must implement an ICMP Echo server function that receives requests sent to the router and sends corresponding replies. The router can ignore ICMP Echo requests addressed to IP broadcast or IP multicast addresses.

### 5.12 Routing processes in UNIX-based systems

This chapter focuses on protocols available in standard IP routers. However, several of these protocols are also available in UNIX-based systems.

These protocols are often implemented using one of two processes:

Routed (pronounced route-D): This is a basic routing process for interior routing. It is supplied with the majority of TCP/IP implementations. It implements the RIP protocol.

Gated (pronounced gate-D): This is a more sophisticated process allowing for both interior and exterior routing. It can implement a number of protocols including OSPF, RIP-2, and BGP-4.

### 5.13 RFCs relevant to this chapter

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

[RFC 904 – Exterior Gateway Protocol formal specification (April 1984)](http://www.ietf.org/rfc/rfc904.txt)

[RFC 1058 – Routing Information Protocol (June 1988)](http://www.ietf.org/rfc/rfc1058.txt)

[RFC 1322 – A Unified Approach to Inter-Domain Routing (May 1992)](http://www.ietf.org/rfc/rfc1322.txt)

[RFC 1812 – Requirements for IP Version 4 Routers. (June 1995)](http://www.ietf.org/rfc/rfc1812.txt)

[RFC 2080 – RIPng for IPv6 (January 1997)](http://www.ietf.org/rfc/rfc2080.txt)

[RFC 2328 – OSPF Version 2 (April 1998)](http://www.ietf.org/rfc/rfc2328.txt)

[RFC 2453 – RIP Version 2 (November 1998)](http://www.ietf.org/rfc/rfc2453.txt)

[RFC 3065 – Autonomous System Confederations for BGP (February 2001)](http://www.ietf.org/rfc/rfc3065.txt)

[RFC 3101 – The OSPF Not-So-Stubby Area (NSSA) Option (January 2003)](http://www.ietf.org/rfc/rfc3101.txt)

[RFC 4271 – A Border Gateway Protocol 4 (BGP-4) (January 2006)](http://www.ietf.org/rfc/rfc4271.txt)

[RFC 4451 – BGP MULTI\_EXIT\_DISC (MED) Considerations (March 2006)](http://www.ietf.org/rfc/rfc4451.txt)

[RFC 4456 – BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP) (April 2006)](http://www.ietf.org/rfc/rfc4456.txt)

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **12** | |

## Chapter 12. Directory and naming protocols

An inherent problem in making resources available through a network is creating an easy way of accessing these resources. On small networks, it might be easy enough to simply remember or write down the information needed to remotely access resources. However, this solution does not scale as networks, and the number of available resources, continue to grow. This becomes increasingly more complex as resources are made available outside of individual networks, to multiple networks, or even across the Internet, on multiple platforms and across a variety of differing hardware. To overcome this, directory and naming methods were devised to provide a uniform method of obtaining the information needed to access a networked resource. This chapter reviews four of these methods:

Domain Name System (DNS)

Dynamic Domain Name System (DDNS)

Network Information System (NIS)

Lightweight Directory Access Protocol (LDAP)

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

### 12.1 Domain Name System (DNS)

The Domain Name System is a standard protocol with STD number 13, and its status is recommended. It is described in RFC 1034 and RFC 1035. This section explains the implementation of the Domain Name System and the implementation of name servers.

The early Internet configurations required the use of only numeric IP addresses. Because this was burdensome and much harder to remember than just the name of a system, this evolved into the use of symbolic host names. For example, instead of typing: TELNET 10.12.7.14

You can type:

TELNET MyHost

MyHost is then translated in some way to the IP address 10.12.7.14. Though using host names makes the process of accessing a resource easier, it also introduces the problem of maintaining the mappings between IP addresses and high-level machine names in a coordinated and centralized way.

Initially, host names to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT), which was fetched by all hosts using FTP. This is called a *flat namespace*. But due to the explosive growth in the number of hosts, this mechanism became too cumbersome (consider the work involved in the addition of just one host to the Internet) and was replaced by a new concept: *Domain Name System*. Hosts on smaller networks can continue to use a local flat namespace (the HOSTS.LOCAL file) instead of or in addition to the Domain Name System. Outside of small networks, however, the Domain Name System is essential. This system allows a program running on a host to perform the mapping of a high-level symbolic name to an IP address for any other host without requiring every host to have a complete database of host names.

#### 12.1.1 The hierarchical namespace

Consider the typical internal structure of a large organization. Because the chief executive cannot do everything, the organization will probably be partitioned into divisions, each of them having autonomy within certain limits. Specifically, the executive in charge of a division has authority to make direct decisions, without permission from the chief executive.

Domain names are formed in a similar way, and will often reflect the hierarchical delegation of authority used to assign them. For example, consider the name: myHost.myDept.myDiv.myCorp.com

In this example, we know that there is a single host name *myHost*, which exists within the *myDept.myDiv.myCorp* subdomain. The *myDept.myDiv.myCorp* subdomain is one of the subdomains of *myDiv.myCorp.com* subdomain, which is in turn one of the subdomains of *myCorp.com.* Finally, *myCorp.com* is a subdomain of *com.* This hierarchy is better illustrated in Figure 12-1.

(

root

)

du

e

ov

g

mil

m

co

Pentagon

RPA

DA

Corp

my

SF

N

OT

D

me

Na

Ti

tle

Name

Title

my

Di

v

y

ourDiv

yourDept

myDept

*Figure 12-1 DNS hierarchical namespace*

We discuss this hierarchical structure at greater length in the following sections.

**12.1.2**  **Fully qualified domain names (FQDNs)**  When using the Domain Name System, it is common to work with only a part of the domain hierarchy, such as the myDivision.myCorp.com domain. The Domain Name System provides a simple method of minimizing the typing necessary in this circumstance. If a domain name ends in a dot (for example,

myDept.myDiv.myCorp.com.), it is assumed to be complete. This is called a *fully qualified domain name (FQDN)* or an *absolute domain name*. However, if it does not end in a dot (for example, myDept.myDiv), it is incomplete and the DNS resolver may complete this by appending a suffix such as .myCorp.com to the domain name. The rules for doing this are implementation-dependent and locally configurable.

#### 12.1.3 Generic domains

The top-level names are called the generic top-level domains (gTLDs), and can be three characters or more in length. Table 12-1 shows some of the top-level domains of today's Internet domain namespace.

*Table 12-1 Current generic domains*

|  |  |
| --- | --- |
| **Domain name** | **Meaning** |
| aero | The air transport industry |
| biz | Business use |
| cat | The Catalan culture |
| com | Commercial organizations |
| coop | Cooperatives |
| edu | Educational organizations |
| gov | U.S. governmental agencies |
| info | Informational sites |
| int | International organizations |
| jobs | Employment-related sites |
| mil | The U.S. military |
| mobi | Mobile devices sites |
| museum | Museums |
| **Domain name** | **Meaning** |
| name | Family and individual sites |
| net | Network infrastructures |
| org | Non-commercial organizations |
| pro | Professional sites |
| travel | The travel industry |

These names are registered with and maintained by the Internet Corporation for Assigned Names and Numbers (ICANN). For current information, see the ICANN Web site at:

[http://www.icann.org](http://www.icann.org/)

#### 12.1.4 Country domains

There are also top-level domains named for the each of the ISO 3166 international 2-character country codes (from *ae* for the United Arab Emirates to *zw* for Zimbabwe). These are called the *country* domains or the *geographical* domains. Many countries have their own second-level domains underneath which parallel the generic top-level domains. For example, in the United Kingdom, the domains equivalent to the generic domains .*com* and .*edu* are .*co*.*uk* and .*ac*.*uk* (ac is an abbreviation for academic). There is a .us top-level domain, which is organized geographically by state (for example, .ny.us refers to the state of New York). See RFC 1480 for a detailed description of the .us domain.

#### 12.1.5 Mapping domain names to IP addresses

The mapping of names to addresses consists of independent, cooperative systems called name servers. A name server is a server program that holds a master or a copy of a name-to-address mapping database, or otherwise points to a server that does, and that answers requests from the client software, called a name resolver.

Conceptually, all Internet domain servers are arranged in a tree structure that corresponds to the naming hierarchy in Figure 12-1 on page 427. Each leaf represents a name server that handles names for a single subdomain. Links in the conceptual tree do not indicate physical connections. Instead, they show which other name server a given server can contact.

#### 12.1.6 Mapping IP addresses to domain names: Pointer queries

The Domain Name System provides for a mapping of symbolic names to IP addresses and vice versa. While the hierarchical structure makes it easy in principle to search the database for an IP address using its symbolic name, the

process of mapping an IP address to a symbolic name cannot use the same process. Therefore, there is another namespace that facilitates the reverse mapping of IP address to symbolic name. It is found in the domain in-addr.arpa (arpa is used because the Internet was originally the *ARPAnet*).

Not including IPv6, IP addresses are normally written in dotted decimal format, and there is one layer of domain for each hierarchy. Contrary to domain names, which have the least-significant parts of the name first, the dotted decimal format has the most significant bytes first. Therefore, in the Domain Name System, the dotted decimal address is shown in reverse order.

For example, consider the following IPv4 address:

129.34.139.30

The in-add.arpa address for this is:

30.139.34.129.in-addr.arpa.

This is handled slightly different for IPv6 addresses. Because of the IPv6 address’ structure, the reverse order is done in nibbles in stead of octets. Also, the in-addr.arpa domain does not include IPv6. Instead, the domain used is IP6.ARPA. For example, consider the following IPv6 address:

4321:0:1:2:3:4:567:89ab

Breaking this into nibbles, reversing the odder, and appending the domain yields:

b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA

Given an IP address, the Domain Name System can be used to find the matching host name. A domain name query to do this is called a *pointer query*.

#### 12.1.7 The distributed name space

The Domain Name System uses the concept of a *distributed name space*. Symbolic names are grouped into *zones of authority*, more commonly referred to as *zones*. In each of these zones, one or more hosts has the task of maintaining a database of symbolic names and IP addresses within that zone, and provides a server function for clients who want to translate between symbolic names and IP addresses. These local name servers are then (through the internetwork on which they are connected) logically interconnected into a hierarchical tree of *domains*. Each zone contains a part or a *subtree* of the hierarchical tree, and the names within the zone are administered independently of names in other zones. Authority over zones is vested in the name servers.

Normally, the name servers that have authority for a zone will have domain names belonging to that zone, but this is not required. Where a domain contains a subtree that falls in a different zone, the name server or servers with authority over the superior domain are said to *delegate authority* to the name server or servers with authority over the subdomain. Name servers can also delegate authority to themselves; in this case, the domain name space is still divided into zones moving down the domain name tree, but authority for two zones is held by the same server. The division of the domain name space into zones is accomplished using resource records stored in the Domain Name System.

At the top-level root domain there is an exception to this. There is no higher system to which authority can be delegated, but it is not desirable to have all queries for fully qualified domain names to be directed to just one system. Therefore, authority for the top-level zones is shared among a set of *root name servers*[[3]](#footnote-3) coordinated by the ICANN.

To better illustrate the process of resolving a symbolic name to an IP address, consider a query for myHost.myDept.myCorp.com, and let us assume that our name server does not have the answer already in its cache. The query goes to the .com root name server, which in turn forwards the query to a server with an NS record for myCorp.com. At this stage, it is likely that a name server has been reached that has cached the needed answer. However, the query could be further delegated to a name server for myDept.myCorp.com

As a result of this scheme:

Rather than having a central server for the database, the work that is involved in maintaining this database is off-loaded to hosts throughout the name space.

Authority for creating and changing symbolic host names and responsibility for maintaining a database for them is delegated to the organization owning the zone (within the name space) containing those host names.

From the user's standpoint, there is a single database that deals with these address resolutions. The user might be aware that the database is distributed, but generally need not be concerned about this.

**Note:** Although domains within the namespace will frequently map in a logical fashion to networks and subnets within the IP addressing scheme, this is not a requirement of the Domain Name System. Consider a router between two subnets. It has two IP addresses, one for each network adapter, but it would not normally have two symbolic names.

#### 12.1.8 Domain name resolution

The domain name resolution process can be summarized in the following steps:

1. A user program issues a request such as the **gethostbyname()** system call (this particular call asks for the IP address of a host by passing the host name) or the **gethostname()** system call (which asks for a host name of a host by passing the IP address).
2. The resolver formulates a query to the name server. (Full resolvers have a local name cache to consult first; stub resolvers do not. See “Domain name full resolver” and “Domain name stub resolver” on page 434.)
3. The name server checks to see if the answer is in its local authoritative database or cache, and if so, returns it to the client. Otherwise, it queries other available name servers, starting down from the root of the DNS tree or as high up the tree as possible.
4. The user program is finally given a corresponding IP address (or host name, depending on the query) or an error if the query could not be answered. Normally, the program will not be given a list of all the name servers that have been consulted to process the query.

Domain name resolution is a client/server process (see 11.1.1, “The client/server model” on page 408). The client function (called the *resolver* or *name resolver*) is transparent to the user and is called by an application to resolve symbolic high-level names into real IP addresses or vice versa. The name server (also called a *domain name server*) is the server application providing the translation between high-level machine names and the IP addresses. The query/reply messages can be transported by either UDP or TCP.

##### Domain name full resolver

Figure 12-2 shows a program called a *full resolver*, which is distinct from the user program, that forwards all queries to a name server for processing. Responses are cached by the name server for future use.

User

Program

Full

Resolver

Cache

Name

Server

Foreign

Name

Server

C

a

c

h

e

Database

user

query

user

response

query

response

r

q

*Figure 12-2 DNS: Using a full resolver for domain name resolution*

##### Domain name stub resolver

Figure 12-3 shows a *stub resolver*, a routine linked with the user program, that forwards the queries to a name server for processing. Responses are cached by the name server, but not usually by the resolver, although this is implementation dependent. On most platforms, the stub resolver is implemented by two library routines (or by some variation of these routines): **gethostbyname()** and **gethostbyaddr()**. These are used for converting host names to IP addresses and vice versa. Stub resolvers are much more common than full resolvers.

User

Program

Stub

Resolver

Name

Server

Database

Foreign

Name

Server

C

a

c

h

e

query

response

r

q

*Figure 12-3 DNS: Using a stub resolver for domain name resolution*

##### Domain name resolver operation

Domain name queries can be one of two types: *recursive* or *iterative* (also called *non-recursive*). A flag bit in the domain name query specifies whether the client desires a recursive query, and a flag bit in the response specifies whether the server supports recursive queries. The difference between a recursive and an iterative query arises when the server receives a request for which it cannot supply a complete answer by itself. A recursive query requests that the server issues a query itself to determine the requested information and returns the complete answer to the client. An iterative query means that the name server

returns what information it has available and also a list of additional servers for the client to contact to complete the query.

Domain name responses can be one of two types: *authoritative* and *non-authoritative*. A flag bit in the response indicates which type a response is. When a name server receives a query for a domain in a zone over which it has authority, it returns all of the requested information in a response with the authoritative answer flag set. When it receives a query for a domain over which it does not have authority, its actions depend on the setting of the recursion desired flag in the query:

If the recursion desired flag is set and the server supports recursive queries, it will direct its query to another name server. This will either be a name server with authority for the domain given in the query, or it will be one of the root name servers. If the second server does not return an authoritative answer (for example, if it has delegated authority to another server), the process is repeated.

When a server (or a full resolver program) receives a response, it will cache it to improve the performance of repeat queries. The cache entry is stored for a maximum length of time specified by the originator in a 32-bit *time-to-live (TTL)* field contained in the response. A typical TTL value is 86,400 seconds (one day).

If the recursion desired flag is not set or the server does not support recursive queries, it will return whatever information it has in its cache and also a list of additional name servers to be contacted for authoritative information.

##### Domain name server operation

Each name server has *authority* for zero or more zones. There are three types of

|  |  |
| --- | --- |
| name servers: |  |
| **Primary** | A primary name server loads a zone's information from disk and has authority over the zone. |
| **Secondary** | A secondary name server has authority for a zone, but obtains its zone information from a primary server using a process called *zone transfer*. To remain synchronized, the secondary name servers query the primary on a regular basis (typically three hours) and re-execute the zone transfer if the primary has been updated. A name server can operate as a primary or a secondary name server for multiple domains, or a primary for some domains and as a secondary for others. A primary or secondary name server performs all of the functions of a caching-only name server. |
| **Caching-only** | A name server that does not have authority for any zone is called a caching-only name server. A caching-only name server obtains all of its data from primary or secondary name servers as required. It requires at least one NS record to point to a name server from which it can initially obtain information. |

When a domain is registered with the root and a separate zone of authority established, the following rules apply:

The domain must be registered with the root administrator.

There must be an identified administrator for the domain.

There must be at least two name servers with authority for the zone that are accessible from outside and inside the domain to ensure no single point of failure.

We also recommend that name servers that delegate authority apply these rules, because the delegating name servers are responsible for the behavior of name servers under their authority.

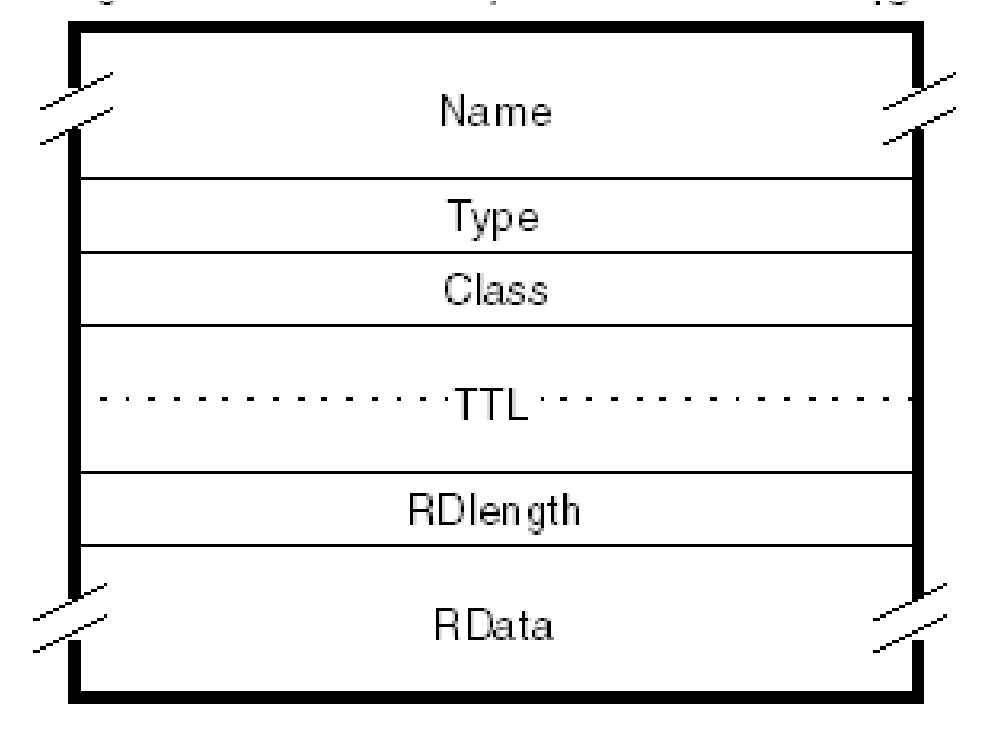
#### 12.1.9 Domain Name System resource records

The Domain Name System's distributed database is composed of *resource records* (RRs), which are divided into classes for different kinds of networks. We only discuss the Internet class of records. Resource records provide a mapping between domain names and *network objects*. The most common network objects are the addresses of Internet hosts, but the Domain Name System is designed to accommodate a wide range of different objects.

A zone consists of a group of resource records, beginning with a Start of

Authority (SOA) record. The SOA record identifies the domain name of the zone. There will be a name server (NS) record for the primary name server for this zone. There might also be NS records for the secondary name servers. The NS records are used to identify which of the name servers are authoritative (see “Domain name resolver operation” on page 434). Following these records are the resource records, which might map names to IP addresses or aliases to names.

The following figure shows the general format of a resource record (Figure 12-4).



*Figure 12-4 DNS general resource record format*

Where:

|  |  |
| --- | --- |
| **Name** | The domain name to be defined. The Domain Name System is very general in its rules for the composition of domain names. However, it recommends a syntax for domain names that minimizes the likelihood of applications that use a DNS resolver (that is, nearly all TCP/IP applications) from misinterpreting a domain name. A domain name adhering to this recommended syntax will consist of a series of labels consisting of alphanumeric characters or hyphens, each label having a length of between 1 and 63 characters, starting with an alphabetic character. Each pair of labels is separated by a dot (period) in human-readable form, but not in the form used within DNS messages. Domain names are not case-sensitive. |
| **Type** | Identifies the type of the resource in this record. There are numerous possible values, but some of the more common ones, along with the RFCs which define them, are listed in Table 12-2 on page 438. |
| **Class** | Identifies the protocol family. The only commonly used value is IN (the Internet system), though other values are defined by  RFC 1035 and include:   * CS (value 2): The CSNET class. This has been obsoleted. * CH (value 3): The CHAOS class. * HS (value 4): The Hesiod class. |
| **TTL** | The time-to-live (TTL) time in seconds for which this resource record will be valid in a name server cache. This is stored in the DNS as an unsigned 32-bit value. A typical value for records pointing to IP addresses is 86400 (one day). |
| **RDlength** | An unsigned 16-bit integer that specifies the length, in octets, of the RData field. |
| **RData** | A variable length string of octets that describes the resource. The format of this information varies according to the Type and Class of the resource record. |

*Table 12-2 Some of the possible resource record types*

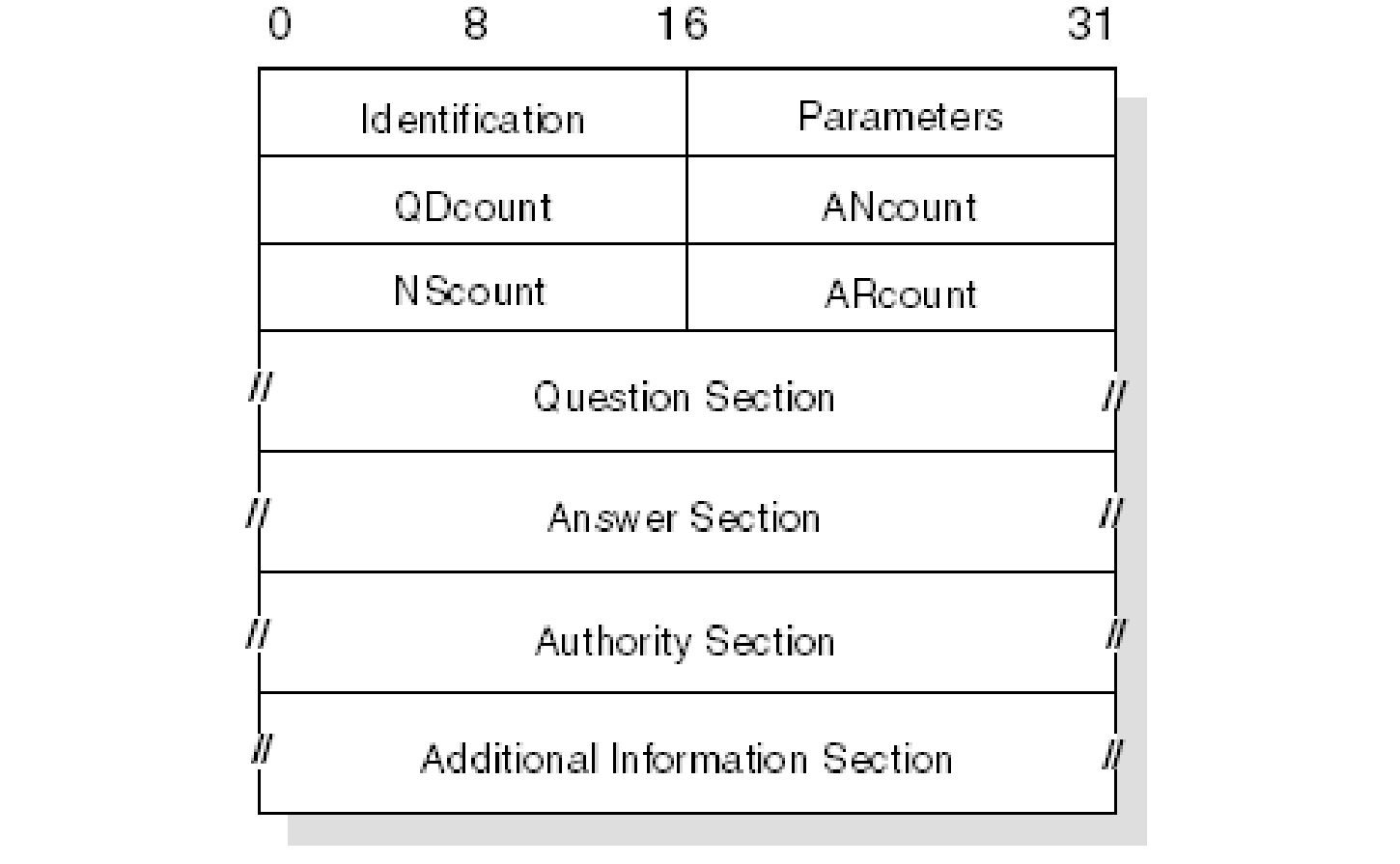
|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Value** | **Meaning** | **RFC def** |
| A | 1 | A host address | 1035 |
| NS | 2 | An authoritative name server | 1035 |
| CNAME | 5 | The canonical name for an alias | 1035 |
| SOA | 6 | Marks the start of a zone of authority | 1035 |
| MB | 7 | A mailbox domain name (experimental) | 1035 |
| MG | 8 | A mail group member (experimental) | 1035 |
| MR | 9 | A mail rename domain name (experimental) | 1035 |
| NULL | 10 | A NULL resource record (experimental) | 1035 |
| WKS | 11 | A well-known service description | 1035 |
| PTR | 12 | A domain name pointer | 1035 |
| HINFO | 13 | Host information | 1035 |
| MINFO | 14 | Mailbox or mail list information | 1035 |
| MX | 15 | Mail exchangea | 1035 |
| TXT | 16 | Text strings | 1035 |
| RP | 17 | Responsible person record | 1183 |
| AFSDB | 18 | Andrew File System database | 1183 |
| X25 | 19 | X.25 resource record | 1183 |
| ISDN | 20 | ISDN resource record | 1183 |
| RT | 21 | Route Through resource record | 1183 |
| NSAP | 22 | Network Service Access Protocol record | 1348 |
| **Type** | **Value** | **Meaning** | **RFC def** |
| NSAP-PTR | 23 | NSAP Pointer record | 1348 |
| KEY | 25 | The public key associated with a DNS name | 2535 |
| AAAA | 28 | An IPv6 address record | 3596 |
| LOC | 29 | GPS resource record | 1876 |
| SRV | 33 | Defines the services available in a zone | 2872 |
| CERT | 37 | Certificate resource records | 4398 |
| A6 | 38 | Forward mapping of an IPv6 address | 2874 |
| DNAME | 39 | Delegation of IPv6 reverse addresses | 2672 |
| DS | 39 | Delegated Signer record (DNS security) | 4034 |
| RRSIG | 46 | Resource record digital signature | 4034 |
| NSEC | 47 | Next Secure record (DNS security) | 4034 |
| DNSKEY | 48 | Public Key record (DNS security | 4034 |

a. The MX Type obsoletes Types MD (value 3, Mail destination) and MF (value 4, Mail forwarder).

#### 12.1.10 Domain Name System messages

All messages in the Domain Name System protocol use a single format. This format is shown in Figure 12-5 on page 440. This frame is sent by the resolver to the name server. Only the header and the question section are used to form the query. Replies and forwarding of the query use the same frame, but with more sections filled in (the answer/authority/additional sections).

*Figure 12-5 DNS message format*



##### Header format

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length.

**ID** A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation of responses when multiple queries are outstanding at the same time.

**Parameters** A 16-bit value in the following format (Table 12-3).

*Table 12-3 Parameters*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 10 | 11 | 12 13 | 14 | 15 |
| Q  R | Op code |  |  | A  A | T  C | R  D | R  A | Zero |  | Rcode |  |  |

Where:

|  |  |
| --- | --- |
| **QR** | Flag identifying a query (0) or a response(1). |
| **Op code** | 4-bit field specifying the kind of query: |

1. Standard query (QUERY)
2. Inverse query (IQUERY)
3. Server status request (STATUS)

Other values are reserved for future use:

|  |  |
| --- | --- |
| **AA** | Authoritative answer flag. If set in a response, this flag specifies that the responding name server is an authority for the domain name sent in the query. |
| **TC** | Truncation flag. Set if message was longer than permitted on the physical channel. |
| **RD** | Recursion desired flag. This bit signals to the name server that recursive resolution is asked for. The bit is copied to the response. |
| **RA** | Recursion available flag. Indicates whether the name server supports recursive resolution. |
| **Zero** | 3 bits reserved for future use. Must be zero. |
| **Rcode** | 4-bit response code. Possible values are:   1. No error. 2. Format error. The server was unable to interpret the message. 3. Server failure. The message was not processed because of a problem with the server. 4. Name error. The domain name in the query does not exist.   This is only valid if the AA bit is set in the response.   1. Not implemented. The requested type of query is not implemented by name server. 2. Refused. The server refuses to respond for policy reasons.   Other values are reserved for future use. |
| **QDcount** | An unsigned 16-bit integer specifying the number of entries in the question section. |
| **ANcount** | An unsigned 16-bit integer specifying the number of RRs in the answer section. |
| **NScount** | An unsigned 16-bit integer specifying the number of name server RRs in the authority section. |
| **ARcount** | An unsigned 16-bit integer specifying the number of RRs in the additional records section. |

##### Question section

The next section contains the queries for the name server. It contains QDcount (usually 1) entries, each in the format shown in Figure 12-6.

*Figure 12-6 DNS question format*

*2*



Where:

|  |  |
| --- | --- |
| **Length** | A single byte giving the length of the next label. |
| **Label** | One element of the domain name characters (for example, ibm from ral.ibm.com). The domain name referred to by the question is stored as a series of these variable length labels, each preceded by a 1-byte length. |
| **00** | X'00' indicates the end of the domain name and represents the null label of the root domain. |
| **Type** | 2 bytes specifying the type of query. It can have any value from the Type field in a resource record. |
| **Class** | 2 bytes specifying the class of the query. For Internet queries, this will be IN. |

2 Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format.

For example, the domain name mydiv.mycorp.com is encoded with the following fields:

X'05'

"mydiv"

X'06'

"mycorp"

X'03'

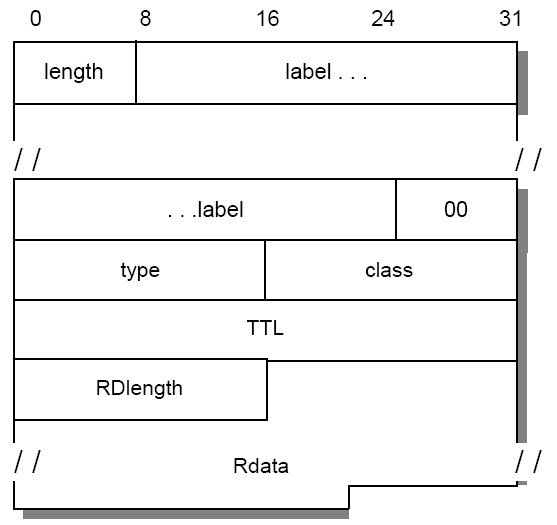
"com"

X'00'

Therefore, the entry in the question section for mydiv.mycorp.com requires 22 bytes: 18 to store the domain name and 2 each for the Qtype and Qclass fields.

##### Answer, authority, and additional resource sections

These three sections contain a variable number of resource records. The number is specified in the corresponding field of the header. The resource records are in the format shown in Figure 12-7.



*Figure 12-7 DNS: Answer Record Entry format [[4]](#footnote-4)*

Where the fields before the TTL field have the same meanings as for a question entry and:

|  |  |
| --- | --- |
| **TTL** | A 32-bit time-to-live value in seconds for the record. This defines how long it can be regarded as valid. |
| **RDlength** | A 16-bit length for the Rdata field. |
| **Rdata** | A variable length string whose interpretation depends on the Type field. |

##### Message compression

In order to reduce the message size, a compression scheme is used to eliminate the repetition of domain names in the various RRs. Any duplicate domain name or list of labels is replaced with a pointer to the previous occurrence. The pointer has the form of a 2-byte field as shown in Figure 12-8.



*Figure 12-8 DNS message compression*

Where:

The first 2 bits distinguish the pointer from a normal label, which is restricted to a 63-byte length plus the length byte ahead of it (which has a value of <64).

The offset field specifies an offset from the start of the message. A zero offset specifies the first byte of the ID field in the header.

If compression is used in an Rdata field of an answer, authority, or additional section of the message, the preceding RDlength field contains the real length after compression is done.

Refer to 12.2, “Dynamic Domain Name System” on page 453 for additional message formats.

##### Using the DNS Uniform Resource Identifiers (URI)

A DNS can also be queried using a Uniform Resource Identifier This is defined in RFC 4501. Strings are not case sensitive, and adhere to the following format:

“dns:” + [ “//” + dnsauthority + “:” + port + ”/” ] + dnsname +

[ “?” + dnsquery ]

|  |  |  |
| --- | --- | --- |
| Where: |  | |
| **dnsauthority** | The DNS server to which the query should be sent. If this is left blank, the query is sent to the default DNS server. | |
| **dnsname** | The name or IP address to be queried. |
| **dnsquery** | The type of the query to be performed. This can be any combination, separated by a semicolon (;), of: |

**CLASS** Usually IN for internet, the class of the query **TYPE** The type of resource record desired

For example, a request using the URI to resolve www.myCorp.com to an IP address might appear as follows: dns:www.mycorp.com

Additionally, the same request can be sent to the server at 10.1.2.3 on port 5353 using the following: dns://10.1.2.3:5353/www.mycorp.com

Finally, this same query can be made specifying a CLASS of IN and a TYPE of A:

dns://10.1.2.3:5353/www.mycorp.com?class=IN;type=A

#### 12.1.11 A simple scenario

Consider a stand-alone network (no outside connections), consisting of two physical networks:

One has an Internet network address of 129.112.

One has a network address of 194.33.7.

They are interconnected by an IP gateway (VM2). See Figure 12-9for more details.



*Figure 12-9 DNS: A simple configuration: Two networks connected by an IP gateway*

Assume the name server function has been assigned to VM1. Remember that the domain hierarchical tree forms a logical tree, completely independent of the physical configuration. In this simple scenario, there is only one level in the domain tree, which will be referred to as *test.example*.

The zone data for the name server appears as shown in Figure 12-10 and continued in Figure 12-11 on page 448.

;note: an SOA record has no TTL field

;

$origin test.example. ;note 1 @ IN SOA VM1.test.example. ADM.VM1.test.example.

(870611 ;serial number for data

1800 ;secondary refreshes every 30 mn

300 ;secondary reties every 5 mn

604800 ;data expire after 1 week

86400) ;minimum TTL for data is 1 week

;

@ 99999 IN NS VM1.test.example. ;note 2 ;

VM1 99999 IN A 129.112.1.1 ;note 3

99999 IN WKS 129.112.1.1 TCP (SMTP ;note 4

FTP

TELNET

NAMESRV)

;

RT1 99999 IN A 129.112.1.2

IN HINFO IBM RT/PC-AIX ;note 5

RT2 99999 IN A 129.112.1.3

IN HINFO IBM RT/PC-AIX

PC1 99999 IN A 129.112.1.11

PC2 99999 IN A 194.33.7.2

PC3 99999 IN A 194.33.7.3

;

;VM2 is an IP gateway and has 2 different IP addresses

;

VM2 99999 IN A 129.112.1.4

99999 IN A 194.33.7.1

99999 IN WKS 129.112.1.4 TCP (SMTP FTP) IN HINFO IBM-3090-VM/CMS

*Figure 12-10 Zone data for the name server, continued in Figure 12-11 on page 448*

|  |
| --- |
| ;  4.1.112.129.in-addr.arpa. IN PTR VM2 ;note 6  ;  ;Some mailboxes  ;  central 10 IN MX VM2.test.example. ;notes 7 and 8  ;  ;a second definition for the same mailbox, in case VM2 is down  ; central 20 IN MX VM1.test.example. waste 10 IN MX VM2.test.example. |

*Figure 12-11 Zone data for the name server, continued from Figure 12-10 on page 447*

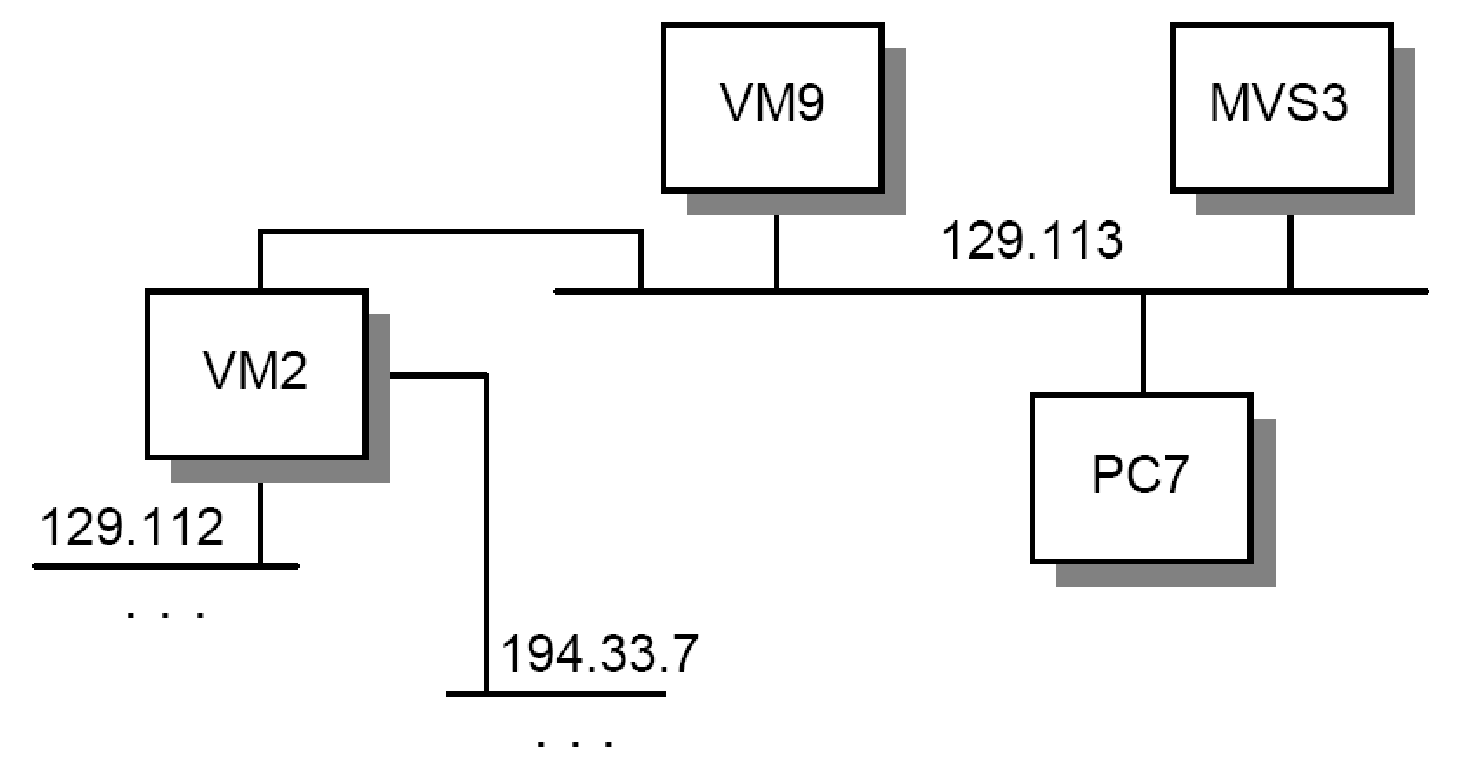
|  |
| --- |
| **Notes for Figure 12-10 on page 447 and Figure 12-11:**   1. The $origin statement sets the @ variable to the zone name   (test.example.). Domain names that do not end with a period are suffixed with the zone name. Fully qualified domain names (those ending with a period) are unaffected by the zone name.   1. Defines the name server for this zone. 2. Defines the Internet address of the name server for this zone. 3. Specifies well-known services for this host. These are expected to always be available. 4. Gives information about the host. 5. Used for inverse mapping queries (see 12.1.6, “Mapping IP addresses to domain names: Pointer queries” on page 430). 6. Will allow mail to be addressed to user@central.test.example. 7. See 15.1.2, “SMTP and the Domain Name System” on page 565 for the use of these definitions. |

#### 12.1.12 Extended scenario

Consider the case where a connection is made to a third network (129.113),

which has an existing name server with authority for that zone (see

Figure 12-12).



*Figure 12-12 DNS: Extended configuration - Third network connected to existing configuration*

Let us suppose that the domain name of the other network is tt.ibm.com and that its name server is located in VM9. All we have to do is add the address of this name server to our own name server database (in the named.ca initial cache file) and reference the other network by its own name server. The following two lines are all that is needed to do that:

tt.ibm.com. 99999 IN NS VM9.tt.ibm.com. VM9.tt.ibm.com. 99999 IN A 129.13.1.9

This simply indicates that VM9 is the authority for the new network and that all queries for that network will be directed to that name server.

#### 12.1.13 Transport

Domain Name System messages are transmitted either as datagrams (UDP) or through stream connection (TCP):

UDP usage: Server port 53 (decimal).

Messages carried by UDP are restricted to 512 bytes. Longer messages are truncated and the truncation (TC) bit is set in the header. Because UDP frames can be lost, a retransmission strategy is required.

TCP usage: Server port 53 (decimal).

In this case, the message is preceded by a 2-byte field indicating the total message frame length.

STD 3 – Host Requirements requires that:

* A Domain Name System resolver or server that is sending a non-zone-transfer query *must* send a UDP query first. If the answer section of the response is truncated and if the requester supports TCP, it tries the query again using TCP. UDP is preferred over TCP for queries because UDP queries have much lower overall processing cost, and the use of UDP is essential for a heavily loaded server. Truncation of messages is rarely a problem given the current contents of the Domain Name System database, because typically 15 response records can be accommodated in the datagram, but this might change as new record types continue to be added to the Domain Name System.
* TCP must be used for zone transfer activities because the 512-byte limit for a UDP datagram will always be inadequate for a zone transfer.
* Name servers must support both types of transport.

As IPv6 becomes more pervasive throughout the Internet community, some problems are forecasted as a result of mixed IPv4/IPv6 network segments. Primarily, if a resolver that can only use IPv4 is forwarded to across a network segment that supports only IPv6, the resolver and the name server will be unable to communicate. As a result, the hierarchical namespace becomes fragmented into two sets of segments: those that support IPv4 and IPv6, and those that support only IPv6. This impending issue has been named the Problem of Name Space Fragmentation, and documented in RFC 3901. In order to preserve namespace continuity, RFC 3901 recommends the following:

Every recursive name server should be either IPv4 only, or dual IPv4 and IPv6.

Every DNS zone should be served by at least one IPv4-reachable authoritative name server.

Additional suggestions about configuring IPv6 DNS servers are in RFC 4339.

##### Dynamic DNS (DDNS)

The Dynamic Domain Name System (DDNS) is a protocol that defines extensions to the Domain Name System to enable DNS servers to accept requests to add, update, and delete entries in the DNS database dynamically. Because DDNS offers a functional superset to existing DNS servers, a DDNS server can serve both static and dynamic domains at the same time, a welcome feature for migration and overall DNS design.

DDNS is currently available in a non-secure and a secure flavor, defined in RFC

2136 and RFC 3007, respectively. Rather than allowing any host to update its DNS records, the secure version of DDNS uses public key security and digital signatures to authenticate update requests from DDNS hosts.

Without client authentication, another host could impersonate an unsuspecting host by remapping the address entry for the unsuspecting host to that of its own. After the remapping occurs, important data, such as logon passwords and mail intended for the host would, unfortunately, be sent to the impersonating host instead.

See 12.2, “Dynamic Domain Name System” on page 453for more information about how DDNS works together with DHCP to perform seamless updates of reverse DNS mapping entries, and see 9.4, “DNS in IPv6” on page 367for more information about DNS with IPv6.

#### 12.1.14 DNS applications

Three common utilities for querying name servers are provided with many DNS implementations: **host**, **nslookup**, and **dig**. Though specifics details about each utility vary slightly by platform, most platforms provide a common set of options.

##### host

The **host** command obtains an IP address associated with a host name, or a host name associated with an IP address. The typical syntax for **host** command is: host [options] name [server]

Where:

**options** Valid options typically include:

**-c class** The query class. By default, this is IN (Internet), but other valid class names include CS (CSNET), CH (CHAOS), HS (Hesiod), and ANY (a wildcard encompassing all four classes).

**-r** Disables recursive processing.

|  |  |
| --- | --- |
|  | **-t type** The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).  **-w** Instructs the **host** command to wait forever  for a reply. |
| **name** | The name of the host or the address to be resolved. |
| **server** | The name server to query. |

##### nslookup

The **nslookup** command enables you to locate information about network nodes, examine the contents of a name server database, and establish the accessibility of name servers. The typical syntax for the **nslookup** command is: nslookup [options] [host] [-nameserver] Where:

|  |  |
| --- | --- |
| **options** | These options vary widely by platform. Refer to the documentation for a specific implementation for information about what options are available. |
| **host** | The host name or IP address to be located. |
| **-nameserver** | The name server to which the query is to be directed. |

##### dig

**dig** stands for Domain Internet Groper, and enables you to exercise name servers, gather large volumes of domain name information, and execute simple domain name queries. The typical syntax for the **dig** command is: dig @server [options] [name] [type] [class] [queryopt]

Where:

|  |  |  |
| --- | --- | --- |
| **@server** | The DNS name server to be queried. | |
| **options** | Valid options typically include: | |
| **-b address** | The source IP address of the query-to address. |
|  | **-c class** | The query class. By default, this is IN  (Internet), but other valid class names |
|  | **-f filename** | include CS (CSNET), CH (CHAOS), and HS    (Hesiod).  Causes **dig** to operate in batch mode, and specifies the file from which the batch commands can be found. |

**-p port** Specifies that **dig** should send the query to a port other than well-known DNS port 53.

**-x address** Instructs **dig** to do a reverse lookup on the specified address.

**name** The name of the resource record to be looked up.

**type** The type of query required. This can be any of the standard resource record types (see Table 12-2 on page 438).

### 12.2 Dynamic Domain Name System

The Domain Name System described in 12.1, “Domain Name System (DNS)” on page 426 is a static implementation without recommendations with regard to security. In order to implement DNS dynamically, take advantage of DHCP, and still to be able to locate any specific host by means of a meaningful label (such as its host name), the following extensions to DNS are required:

A method for the host name to address a mapping entry for a client in the domain name server to be updated after the client has obtained an address from a DHCP server

A method for the reverse address to host name mapping to take place after the client obtains its address

Updates to the DNS to take effect immediately, without the need for intervention by an administrator

Updates to the DNS to be authenticated to prevent unauthorized hosts from accessing the network and to stop imposters from using an existing host name and remapping the address entry for the unsuspecting host to that of its own

A method for primary and secondary DNS servers to quickly forward and receive changes as entries are being updated dynamically by clients

However, implementation of a Dynamic Domain Name System (DDNS) can introduce problems if the environment is not secure. One method of security employed by DNS is the use of Secret Key Transaction Authentication (TSIG), defined in RFC 2845. This can be used to authenticate dynamic updates from clients, or authenticate responses coming from a recursive server. Additionally, these messages can now be protected for integrity and confidentiality through using TSIG over the Generic Security Service (GSS-TSIG). This extension, and the associated algorithms needed to implement GSS-TSIG, are defined in RFC 3645.

In addition to TSIG, and GSS-TSIG, several RFCs extended the functionality of DNS such that it incorporated additional security methods. These additions, defined in RFC 4033 and referred to as the DNS Security Extensions (DNSSEC), allow DNS to authenticate the origin of data as well as negative responses to DNS queries. However, they do not provide confidentiality, access control lists, or protection against denial-of-service-attacks. New resource records relating to security were added by RFCs 4034 and 4398, and include:

DNSKEY (public key)

DS (delegation signer)

RRSIG (resource record digital signature)

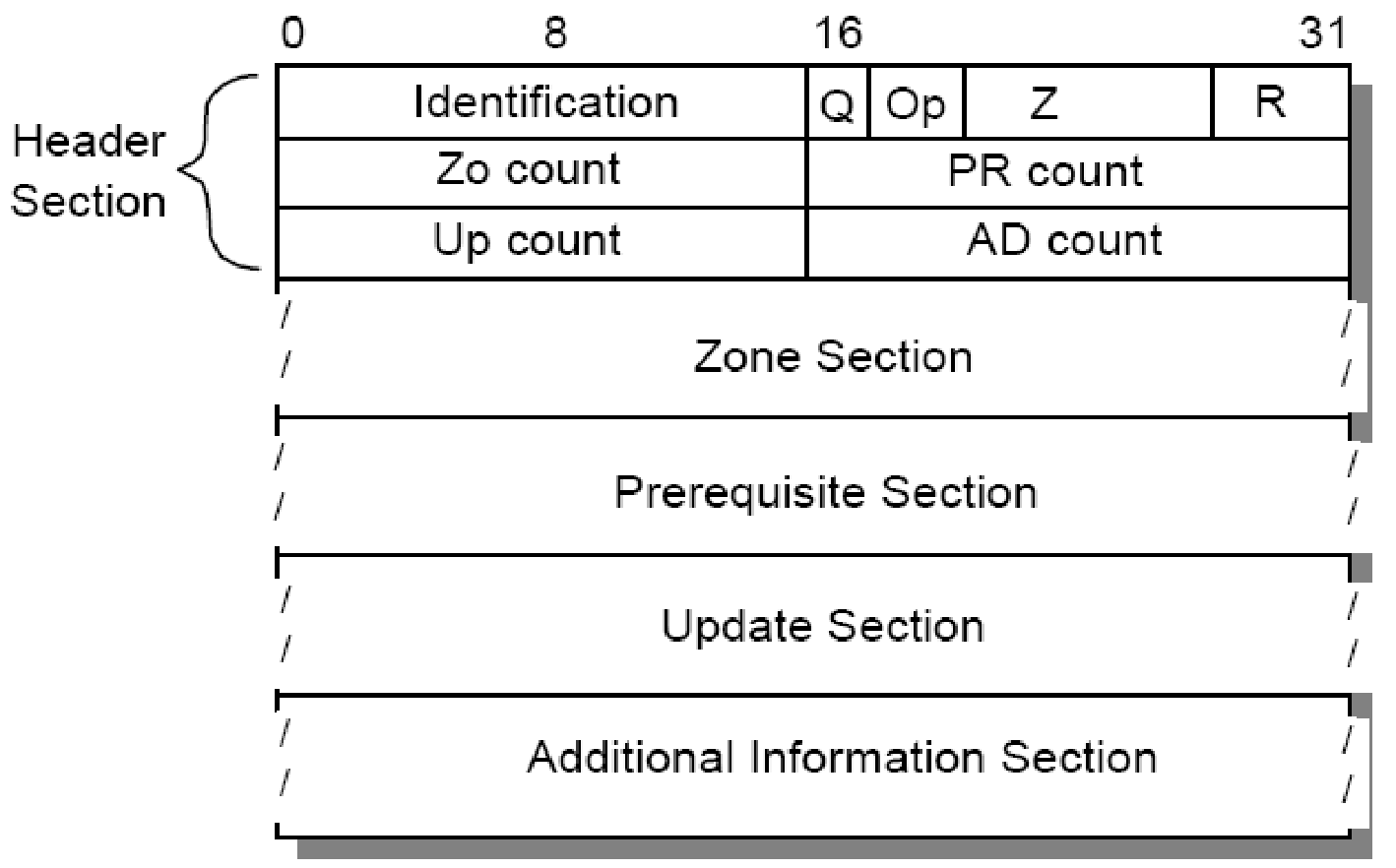
NSEC (authenticated denial of existence)

CERT (public key certificates)

Note that these RRs are also listed in Table 12-2 on page 438. Specific details about how the DNS protocol was modified to take advantage of these additions is in RFC 4035.

#### 12.2.1 Dynamic updates in the DDNS

The DNS message format (shown in Figure 12-5 on page 440) was designed for the querying of a static DNS database. RFC 2136 defines a modified DNS message for updates, called the UPDATE DNS message, illustrated in Figure 12-13 on page 455. This message adds or deletes resource records in the DNS, and allows updates to take effect without the DNS having to be reloaded.



*Figure 12-13 DDNS UPDATE message format*

The header section is always present and has a fixed length of 12 bytes. The other sections are of variable length. They are:

**Identification** A 16-bit identifier assigned by the program. This identifier is copied in the corresponding reply from the name server and can be used for differentiation when multiple queries/updates are outstanding at the same time.

1. Flag identifying an update request (0) or a response (1).

**Op** Opcode. The value 5 indicates an UPDATE message. **z** 7-bit field set to 0 and reserved for future use.

1. Response code (undefined in update requests). Possible

values are:

* 1. No error.
  2. Format error. The server was unable to interpret the message.
  3. Server failure. The message was not processed due to a problem with the server. **3** Name error. A name specified does not exist.

* + 1. Not implemented. The type of message specified in Opcode is not supported by this server.
    2. Refused. The server refuses to perform the UPDATE requested for security or policy reasons.
    3. Name error. A name exists when it should not.
    4. RRset error. A resource record set exists when it should not.
    5. RRset error. A resource record set specified does not exist.
    6. Zone Authority error. The server is not authoritative for the zone specified.
    7. Zone error. A name specified in the Prerequisite or Update sections is not in the zone specified.

|  |  |
| --- | --- |
| **ZO count** | The number of RRs in the Zone section. |
| **PR count** | The number of RRs in the Prerequisite section. |
| **UP count** | The number of RRs in the Update section. |
| **AD count** | The number of RRs in the Additional information section. |
| **Zone section** | This section is used to indicate the zone of the records that are to be updated. As all records to be updated must belong to the same zone, the zone section has a single entry specifying the zone name, zone type (which must be SOA), and zone class. |

##### Prerequisite section

This section contains RRs or RRsets that either must, or must not, exist, depending on the type of update.

**Update section** This section contains the RRs, RRsets, or both that are to be added to or deleted from the zone.

##### Additional information section

This section can be used to pass additional RRs that relate to the update operation in process.

For further information about the UPDATE message format, refer to RFC 2136.

#### 12.2.2 Incremental zone transfers in DDNS

RFC 1995 introduces the IXFR DNS message type, which allows incremental transfers of DNS zone data between primary and secondary DNS servers. In other words, when an update has been made to the zone data, only the change has to be copied to the other DNS servers that maintain a copy of the zone data, rather than the whole DNS database (as is the case with the AXFR DNS message type).

The format of an IXFR query is exactly that of a normal DNS query, but with a query type of IXFR. The response’s answer section, however, is made up of *difference sequences*.

Each list difference sequences is preceded by the server’s current version of the SOA and represents one update to the zone. Similarly, each difference sequence is preceded by an SOA version (indicating in which versions correspond to each change, and the difference sequences are ordered oldest to newest. Upon receiving this message, a server can update its zone by tracking the version history listed in the IXFR answer section.

For example, assume a server has the following zone:

|  |  |
| --- | --- |
| MYZONE.MYDIV.MYCORP | IN SOA MYHOST.MYDIV.MYCORP (  1 600 600 3600000 614800)  IN NS MYHOST.MYDIV.MYCORP |
| MYHOST.MYDIV.MYCORP | IN A 10.1.2.3 |
| OTHERHOST.MYDIV.MYCORP | IN A 10.2.3.4 |

Otherhost.mydiv.mycorp is removed, and in version 2, thishost.mydiv.mycorp is added, leaving the zone as:

MYZONE.MYDIV.MYCORP IN SOA MYHOST.MYDIV.MYCORP (

2 600 600 3600000 614800)

IN NS MYHOST.MYDIV.MYCORP

MYHOST.MYDIV.MYCORP IN A 10.1.2.3

THISHOST.MYDIV.MYCORP IN A 10.2.3.5

If the server receives an IXFR query, it sends back the following answer section:

|  |  |
| --- | --- |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=1 |
| OTHERHOST.MYDIV.MYCORP | IN A 10.1.2.4 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |
| THISHOST.MYDIV.MYCORP | IN A 10.2.3.5 |
| MYZONE.MYDIV.MYCORP | IN SOA serial=2 |

**Note:** If a server received an IXFR query, but incremental zone transfers are not available, it will send back the entire zone in the reply.

#### 12.2.3 Prompt notification of zone transfer

RFC 1996 introduces the NOTIFY DNS message type, which is used by a master server to inform subordinate servers that an update has taken place and that they should initiate a query to discover the new data. The NOTIFY message uses the DNS message format, but only a subset of the available fields (unused

fields are filled with binary zeros). The message is similar to a QUERY message, and can contain the name of the RRs that have been updated. Upon receipt of a NOTIFY message, the subordinate returns a response. The response contains no useful information, and only serves to alert the master server of receipt of the NOTIFY. Based on the RRs contained in the notify, subordinate servers might then send an update query to the server to obtain the new changes.

### 12.3 Network Information System (NIS)

The Network Information System (NIS) is not an Internet standard. It was developed by Sun Microsystems, Inc. It was originally known as the Yellow Pages (YP) and many implementations continue to use this name.

NIS is a distributed database system that allows the sharing of system information in UNIX-based environment. Examples of system information that can be shared include the /etc/passwd, /etc/group, and /etc/hosts files. NIS has the following advantages:

Provides a consistent user ID and group ID name space across a large number of systems

Reduces the time and effort by users in managing their user IDs, group IDs, and NFS file system ownerships

Reduces the time and effort by system administrators in managing user IDs, group IDs, and NFS ownerships

NIS is built on the RPC, and employs the client/server model. Most NIS implementations use UDP. However, because it uses RPC, it is also possible for it to be implemented over TCP. A NIS domain is a collection of systems consisting of:

**NIS master server** Maintains *maps*, or databases, containing the

system information, such as passwords and host names. These are also referred to as Database Maps (DBMs).

**NIS subordinate server(s)** Can be defined to offload the processing from the master NIS server or when the NIS master server is unavailable.

**NIS client(s)** The remaining systems that are served by the

NIS servers.

The NIS clients do not maintain NIS maps; they query NIS servers for system information. Any changes to an NIS map is done only to the NIS master server (through RPC). The master server then propagates the changes to the NIS subordinate servers.

Note that the speed of a network determines the performance and availability of the NIS maps. When using NIS, the number of subordinate servers should be tuned in order to achieve these goals.

Because NIS is not standardized by the IETF, implementations vary by platform.

However, most platforms make available the following common NIS commands:

|  |  |
| --- | --- |
| **makedbm** | Generate a DBM file from an input file. |
| **ypcat** | Display the contents of a DBM file. |
| **ypinit** | Set up an NIS master or subordinate server. |
| **ypmake** | Performs the same function as **makedbm**, but provides the option to push the resulting DBMs to subordinate servers. |
| **ypmatch** | Prints the values associated with one or more keys in a DBM. |
| **yppasswd** | Change a login password stored in a DBM. |
| **yppush** | Pushes DBMs to subordinate servers. |
| **ypwhich** | Indicates what NIS server a client is using. |
| **ypxfr** | Pulls a DBM from the master server. |

### 12.4 Lightweight Directory Access Protocol (LDAP)

When implementing a Distributed Computing Environment (DCE), directory services are automatically included because they are an intrinsic part of the DCE architecture. However, though widely used, implementation of a DCE is not a practical solution for every company needing directory services because it is an “all-or-nothing” architecture. As such, if the other services provided by a DCE are not required, or if implementation of the DCE model is not feasible (for example, if it is not feasible to install the client software on every workstation within the network), other directory service alternatives must be identified.

One such alternative is the Lightweight Directory Access Protocol (LDAP), which is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory, and is gaining wide acceptance as the directory access method of the Internet. It is supported by a growing number of software vendors and is being incorporated into a growing number of applications.

For further information about LDAP, refer to the IBM Redbook *Understanding*  *LDAP - Design and Implementation*, SG24-4986.

#### 12.4.1 LDAP: Lightweight access to X.500

The OSI directory standard, X.500, specifies that communication between the directory client and the directory server uses the Directory Access Protocol

(DAP). However, as an application layer protocol, DAP requires the entire OSI protocol stack to operate, which requires more resources than are available in many small environments. Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol was desired.

LDAP was developed as a *lightweight* alternative to DAP, because it requires the more popular TCP/IP protocol stack rather than the OSI protocol stack. LDAP also simplifies some X.500 operations and omits some esoteric features. Two precursors to LDAP appeared as RFCs issued by the IETF, RFC 1202 – Directory Assistance Service and RFC 1249 – DIXIE Protocol Specification.

These were both informational RFCs which were not proposed as standards. The directory assistance service (DAS) defined a method by which a directory client communicates to a proxy on an OSI-capable host, which issues X.500 requests on the client's behalf. DIXIE is similar to DAS, but provides a more direct translation of the DAP.

The first version of LDAP was defined in RFC 1487 – X.500 Lightweight Access, which was replaced by RFC 1777 – Lightweight Directory Access Protocol. Much of the work on DIXIE and LDAP was carried out at the University of Michigan, which provides reference implementations of LDAP and maintains LDAP-related Web pages and mailing lists. Since then, LDAPv2 has been replaced by LDAP Version 3. LDAPv3 is summarized in RFC 4510, but the technical specifications are divided into multiple subsequent RFCs listed in Table 12-4.

*Table 12-4 LDAP-related RFCs*

|  |  |
| --- | --- |
| **RFC number** | **Content** |
| 4510 | Technical Specification Road Map |
| 4511 | The Protocol |
| 4512 | Directory Information Models |
| 4513 | Authentication Methods and Security Mechanisms |
| 4514 | String Representation of Distinguished Names |
| 4515 | String Representation of Search Filters |
| 4516 | Uniform Resource Locator |
| 4517 | Syntaxes and Matching Rules |
| 4518 | Internationalized String Preparation |
| **RFC number** | **Content** |
| 4519 | Schema for User Applications |
| 4520 | Internet Assigned Numbers Authority (IANA) Considerations for  LDAP |
| 4521 | Considerations for LDAP |
| 4522 | The Binary Encoding Option |
| 4523 | Schema Definitions for X.509 Certificates |
| 4524 | COSINE/ LDAP X.500 Schema |
| 4525 | Modify-Increment Extension |
| 4526 | Absolute True and False Filters |
| 4527 | Read Entry Controls |
| 4528 | Assertion Control |
| 4529 | Requesting Attributes by Object Class in LDAP |
| 4530 | entryUUID Operational Attribute |
| 4531 | Turn Operation |
| 4532 | “Who Am I” Operation |
| 4533 | Content Synchronization Operation |

Though an application program interface (API) for previous versions of LDAP was limited to specifications in RFC 1823, the LDAPv3 provides both a C API and a Java™ Naming and Directory Interface (JNDI).

#### 12.4.2 The LDAP directory server

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. An application client program initiates an LDAP message by calling an LDAP API. But an X.500 directory server does not understand LDAP messages. In fact, the LDAP client and X.500 server even use different communication protocols (TCP/IP versus OSI). The LDAP client actually communicates with a gateway process (also called a proxy or front end) that forwards requests to the X.500 directory server (see Figure 12-14 on page 462), known as an LDAP server, which fulfils requests from the LDAP client. It does this by becoming a client of the X.500 server. The LDAP server must communicate using both TCP/IP (with the client) and OSI (with the X.500 server).

LDAP

Client

LDAP

Server

X.500

Server

Directory

TCP/IP

OSI

*Figure 12-14 LDAP server acting as a gateway to an X.500 directory server*

As the use of LDAP grew and its benefits became apparent, people who did not have X.500 servers or the environments to support them wanted to build directories that could be accessed by LDAP clients. This requires that the LDAP server store and access the directory itself instead of only acting as a gateway to X.500 servers (see Figure 12-15). This eliminates any need for the OSI protocol stack but, of course, makes the LDAP server much more complicated, because it must store and retrieve directory entries. These LDAP servers are often called stand-alone LDAP servers because they do not depend on an X.500 directory server. Because LDAP does not support all X.500 capabilities, a stand-alone LDAP server only needs to support the capabilities required by LDAP.

*Figure 12-15 Stand-alone LDAP server*

LDAP

Client

LDAP

Server

Directory

TCP/IP

The concept of the LDAP server being able to provide access to local directories supporting the X.500 model, rather than acting only as a gateway to an X.500 server, is discussed in RFC 4511 (see Table 12-4 on page 460). From the client's

point of view, any server that implements the LDAP protocol is an LDAP directory server, whether the server actually implements the directory or is a gateway to an X.500 server. The directory that is accessed can be called an LDAP directory, whether the directory is implemented by a stand-alone LDAP server or by an X.500 server.

#### 12.4.3 Overview of LDAP architecture

LDAP defines the content and format of messages exchanged between an LDAP client and an LDAP server. The messages specify the operations requested by the client (search, modify, delete, and so on), the responses from the server, and the format of data carried in the messages. LDAP messages are carried over TCP/IP, a connection-oriented protocol, so there are also operations to establish and disconnect a session between the client and server.

The general interaction between an LDAP client and an LDAP server takes the following form:

1. The client establishes a session with an LDAP server. This is known as *binding* to the server. The client specifies the host name or IP address and TCP/IP port number where the LDAP server is listening. The client can provide a user name and a password to properly authenticate with the server, or the client can establish an anonymous session with default access rights. The client and server can also establish a session that uses stronger security methods, such as encryption of data (see 12.4.5, “LDAP security” on page 471).
2. The client then performs operations on directory data. LDAP offers both read and update capabilities. This allows directory information to be managed as well as queried. LDAP supports searching the directory for data meeting arbitrary user-specified criteria. Searching is the most common operation in LDAP. A user can specify what part of the directory to search and what information to return. A search filter that uses Boolean conditions specifies which directory data matches the search.
3. When the client has finished making requests, it closes the session with the server. This is also known as *unbinding*.

Because LDAP was originally intended as a lightweight alternative to DAP for accessing X.500 directories, the LDAP server follows an X.500 model. The directory stores and organizes data structures known as entries. A directory entry usually describes an object such as a person, a printer, a server, and so on. Each entry has a name called a distinguished name (DN) that uniquely identifies it. The DN consists of a sequence of parts called relative distinguished names (RDNs), much like a file name can consist of a path of directory names. The entries can be arranged into a hierarchical tree-like structure based on their distinguished names. This tree of directory entries is called the directory information tree (DIT).

LDAP defines operations for accessing and modifying directory entries, such as:

Searching for entries meeting user-specified criteria

Adding an entry

Deleting an entry

Modifying an entry

Modifying the distinguished name or relative distinguished name of an entry (move)

Comparing an entry

#### 12.4.4 LDAP models

LDAP can be better understood by considering the four models upon which it is based:

|  |  |
| --- | --- |
| **Information** | Describes the structure of information stored in an LDAP directory. |
| **Naming** | Describes how information in an LDAP directory is organized and identified. |
| **Functional** | Describes the operations that can be performed on the information stored in an LDAP directory. |
| **Security** | Describes how the information in an LDAP directory can be protected from unauthorized access. |

The following sections discuss the first three LDAP models. We describe LDAP security in 12.4.5, “LDAP security” on page 471.

##### The information model

The basic unit of information stored in the directory is an entry, which represents an object of interest in the real world such as a person, server, or organization. Each entry contains one or more attributes that describe the entry. Each attribute has a type and one or more values. For example, the directory entry for a person might have an attribute called telephoneNumber. The syntax of the telephoneNumber attribute specifies that a telephone number must be a string of numbers that can contain spaces and hyphens. The value of the attribute is the person's telephone number, such as 123-456-7890 (a person might have multiple telephone numbers, in which case this attribute would have multiple values).

In addition to defining what data can be stored as the value of an attribute, an attribute syntax also defines how those values behave during searches and other directory operations. This is done using syntax and matching rules. The attribute telephoneNumber, for example, might have a syntax that specifies:

Lexicographic ordering.

Case, spaces, and dashes are ignored during the comparisons. Values must be character strings.

For example, using the correct definitions, the telephone numbers

123-456-7890, 123456-7890, and 1234567890 are considered to be the same. A few of the common syntaxes and matching rules, defined in RFC 4517, are listed in Table 12-5.

*Table 12-5 Examples of LDAP syntaxes*

|  |  |
| --- | --- |
| **Syntaxes and matching rules** | **Description** |
| Bit String | A sequence of binary digits |
| Postal Address | A sequence of strings that form an address in a physical mail system |
| caseExactMatch | A matching rule requiring that string comparisons are case-sensitive |
| caseIgnoreMatch | A matching rule that does not require case-sensitive comparisons |

Table 12-6 lists some common attributes defined by RFC 4519. Some attributes have alias names that can be used wherever the full attribute name is used.

*Table 12-6 Examples of LDAP syntaxes*

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute, alias** | **Syntax** | **Description** | **Example** |
| commonName, cn | cis | Common name of an entry | John Smith |
| surname, sn | cis | A person’s last name | Smith |
| initials | cis | A person’s initials | JS |
| telephoneNumber | tel | A person’s telephone number | 123-456-7890 |

An object class is a general description, sometimes called a template, of an object type, as opposed to the description of a specific object of that type. For example, the object class *person* has a surname attribute, while the object describing John Smith has a surname attribute with the value *Smith*. The object classes that a directory server can store and the attributes they contain are described by a *schema*. A schema defines where object classes are allowed in the directory, which attributes they must contain, which attributes are optional, and the syntax of each attribute. For example, a schema might define a *person* object class that requires that a person has a character-string surname attribute, can optionally have a number-string telephoneNumber attribute, and so on. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schemas also define the inheritance and subclassing of objects, and where in the DIT structure (hierarchy) objects can appear.

Though an implementation can define any schema to meet its needs, RFC 4519 defines a few standard schemas. Table 12-7 lists a few of the common schema (object classes and their required attributes). In many cases, an entry can consist of more than one object class.

*Table 12-7 Examples of object classes and required attributes*

|  |  |  |
| --- | --- | --- |
| **Object class** | **Description** | **Attributes** |
| country | Defines a country | Required: countryCode Optional: searchGuide description |
| locality | Defines a place in the physical world | Required:  None  Optional: street seeAlso searchGuide description |
| person | Defines a person | Required:  surname commonName  Optional:  userPassword telephoneNumber seeAlso description |

Because servers can define their own schema, LDAP includes the functionality of allowing a client to query a server for the contents of the supported schema.

##### The naming model

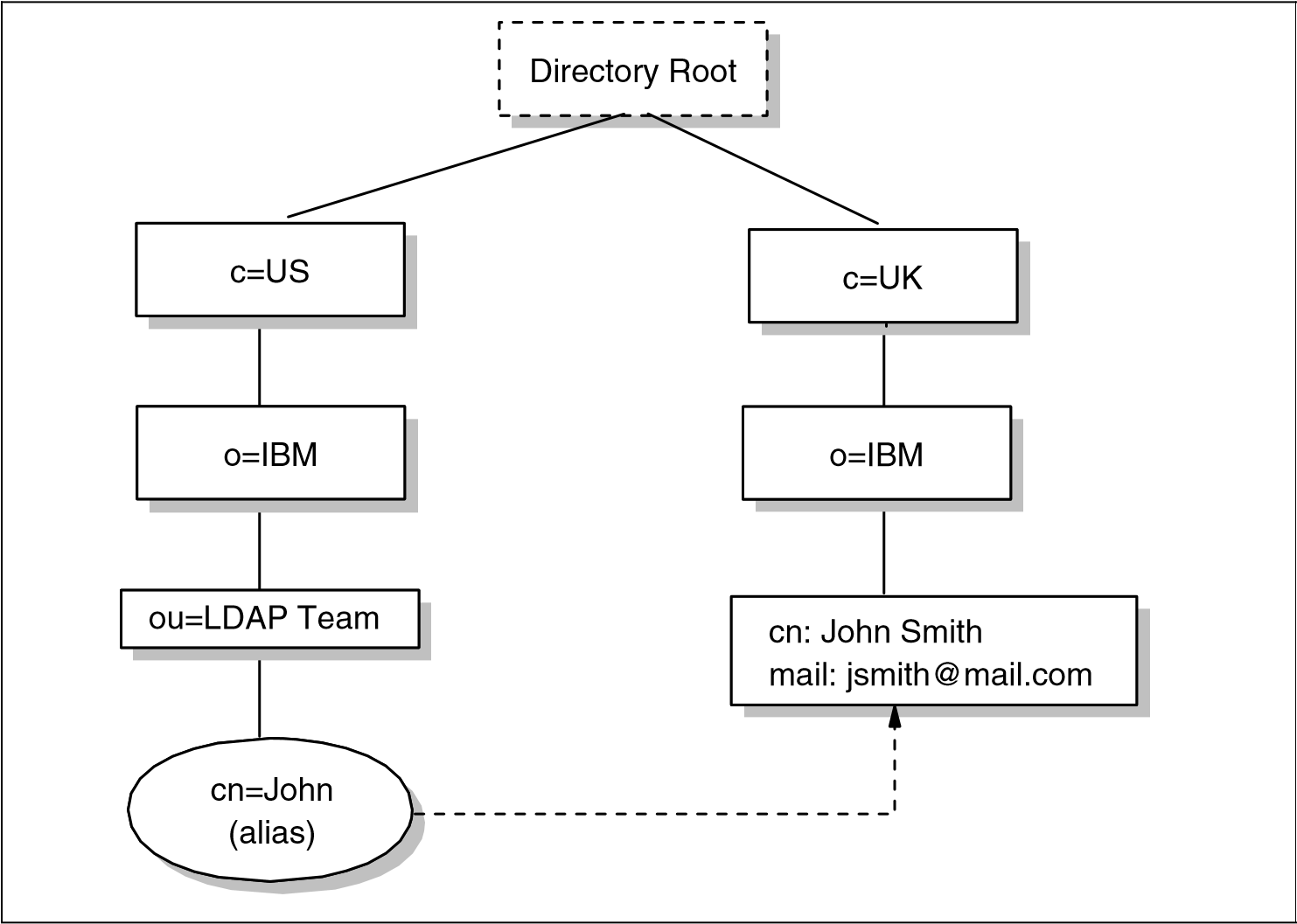
The LDAP naming model defines how entries are identified and organized.

Entries are organized in a tree-like structure called the directory information tree

(DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DNs are made up of a sequence of relative distinguished names (RDNs). Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry.

Each RDN is derived from the attributes of the directory entry. In the simple and common case, an RDN has the form <attribute-name>=<value>. A DN is composed of a sequence of RDNs separated by commas. These relationships are defined in RFC 4514.

An example of a DIT is shown in Figure 12-16. The example is very simple, but can be used to illustrate some basic concepts. Each box represents a directory entry. The root directory entry is conceptual and does not actually exist. Attributes are listed inside each entry. The list of attributes shown is not complete. For example, the entry for the country UK (c=UK) could have an attribute called description with the value United Kingdom.



*Figure 12-16 Example of a directory information tree (DIT)*

It is usual to follow either a geographical or an organizational scheme to position entries in the DIT. For example, entries that represent countries would be at the top of the DIT. Below the countries would be national organizations, states, and provinces, and so on. Below this level, entries might represent people within those organizations or further subdivisions of the organization. The lowest layers of the DIT entries can represent any object, such as people, printers, application servers, and so on. The depth or breadth of the DIT is not restricted and can be designed to suit application requirements.

Entries are named according to their position in the DIT. The directory entry in the lower-right corner of Figure 12-16 on page 467 has the DN cn=John Smith,o=IBM,c=UK.

**Note:** DNs read from leaf-to-root, as opposed to names in a file system directory, which usually read from root-to-leaf.

The DN is made up of a sequence of RDNs. Each RDN is constructed from an attribute (or attributes) of the entry it names. For example, the DN cn=John Smith,o=IBM,c=UK is constructed by adding the RDN cn=John Smith to the DN of the ancestor entry o=IBM,c=UK.

The DIT is described as being tree-like, implying that it is not a tree. This is because of aliases. Aliases allow the tree structure to be circumvented. This can be useful if an entry belongs to more than one organization or if a commonly used DN is too complex. Another common use of aliases is when entries are moved within the DIT and you want access to continue to work as before. In Figure 12-16 on page 467, cn=John,ou=LDAP Team,o=IBM,c=US is an alias for cn=John Smith,o=IBM,c=UK.

Because an LDAP directory can be distributed, an individual LDAP server might not store the entire DIT. Instead, it might store the entries for a particular department but not the entries for the ancestors of the department. For example, a server might store the entries for the Accounting department at Yredbookscorp. The highest node in the DIT stored by the server would be ou=Accounting,o=Yredbookscorp,c=US. The server would store entries ou=Accounting,o=Yredbookscorp,c=US but not for c=US or for o=Yredbookscorp,c=US. The highest entry stored by a server is called a *suffix*. Each entry stored by the server ends with this suffix, so in this case, the suffix is the entire ou=Accounting,o=Yredbookscorp,c=US.

A single server can support multiple suffixes. For example, in addition to storing information about the Accounting department, the same server can store information about the Sales department at MyCorp. The server then has the suffixes ou=Accounting,o=Yredbookscorp,c=US and ou=Sales,o=MyCorp,c=US. Because a server might not store the entire DIT, servers need to be linked together in some way in order to form a distributed directory that contains the entire DIT. This is accomplished with *referrals*. A referral acts as a pointer to an entry on another LDAP server where requested information is stored. A referral is an entry of objectClass *referral*. It has an attribute, *ref*, whose value is the LDAP URL of the referred entry on another LDAP server. See 12.4.6, “LDAP URLs” on page 474 for further information. Referrals allow a DIT to be partitioned and distributed across multiple servers. Portions of the DIT can also be replicated. This can improve performance and availability.

**Note:** When an application uses LDAP to request directory information from a server, but the server only has a referral for that information, the LDAP URL for that information is passed to the client. It is then the responsibility of that client to contact the new server to obtain the information. This is unlike the standard mechanisms of both DCE and X.500, where a directory server, if it does not contain the requested information locally, will always obtain the information from another server and pass it back to the client.

##### The functional model

LDAP defines operations for accessing and modifying directory entries. LDAP operations can be divided into the following three categories:

**Query** Includes the search and compare operations used to

retrieve information from a directory.

**Update** Includes the add, delete, modify, modify RDN, and

unsolicited notification operations used to update stored information in a directory. These operations will normally be carried out by an administrator.

**Authentication** Includes the bind, unbind, abandon, and startTLS operations used to connect and disconnect to and from an LDAP server, establish access rights, and protect information. For further information, see 12.4.5, “LDAP security” on page 471.

###### The search operation

The most common operation is the search. This operation is very flexible and therefore has some of the most complex options. The search operation allows a client to request that an LDAP server search through some portion of the DIT for information meeting user-specified criteria in order to read and list the results.

The search can be very general or very specific. The search operation allows the specification of the starting point within the DIT, how deep within the DIT to search, the attributes an entry must have to be considered a match, and the attributes to return for matched entries.

Some example searches expressed informally are:

Find the postal address for cn=John Smith,o=IBM,c=UK.

Find all the entries that are children of the entry ou=ITSO,o=IBM,c=US.

Find the e-mail address and phone number of anyone in an organization whose last name contains the characters “miller” and who also has a fax number.

To perform a search, the following parameters must be specified:

**Base** A DN that defines the starting point, called the base

object, of the search. The base object is a node within the DIT.

**Scope** Specifies how deep within the DIT to search from the base object. There are three choices: **baseObject** Only the base object is examined.

**singleLevel** Only the immediate children of the base object are examined; the base object itself is not examined.

**wholeSubtree** The base object and all of its

descendants are examined.

**Alias dereferencing** Specifies if aliases are dereferenced. That is, the actual object of interest, pointed to by an alias entry, is examined. Not dereferencing aliases allows the alias entries themselves to be examined. This parameter must be one of the following:

**neverDerefAliases**

Do not deference aliases.

derefInSearching

Dereference aliases only when searching subordinates of the base object.

derefFindingBaseObj

Dereference aliases only when searching for the base object, but not when searching subordinates of the base object.

**derefAlways** Always dereference aliases.

**Size Limit** The maximum number of entries that should be returned as a result of the search.

**Time Limit** The maximum number of seconds allowed to perform the

search. Specifying zero indicates that there is no time limit.

**Types Only** This parameter has two possible values:

|  |  |
| --- | --- |
| **TRUE** | Only attribute descriptions are returned. |
| **FALSE** | Attribute descriptions and values are returned. |

**Search filter** Specifies the criteria an entry must match to be returned from a search. The search filter is a Boolean combination of attribute value assertions. An attribute value assertion tests the value of an attribute for equality, less than or equal, and so on.

**Attributes to return** Specifies which attributes to retrieve from entries that match the search criteria. Because an entry can have many attributes, this allows the user to only see the attributes in which they are interested.

#### 12.4.5 LDAP security

Security is of great importance in the networked world of computers, and this is true for LDAP as well. When sending data over insecure networks, internally or externally, sensitive information might need to be protected during transportation. There is also a need to know who is requesting the information and who is sending it. This is especially important when it comes to the update operations on a directory. RFC 4513 discusses the authentication methods and security mechanisms available in LDAPv3, which can be divided into the following sections:

|  |  |
| --- | --- |
| **Authentication** | Assurance that the opposite party (machine or person) really is who he/she/it claims to be. |
| **Integrity** | Assurance that the information that arrives is really the same as what was sent. |
| **Confidentiality** | Protection against information disclosure, by means of data encryption, to those who are not intended to receive it. |
| **Authorization** | Assurance that a party is really allowed to do what it is requesting to do, usually checked after user  authentication. Authorization is achieved by assigning access controls, such as read, write, or delete, for user IDs or common names to the resources being accessed. Because these attributes are usually platform-specific,  LDAP does not provide specific controls. Instead, it has |

built-in mechanisms to allow the use of the platform-specific controls.

Because the use of authorization controls is platform-specific, the following sections describe only the authentication, integrity, and confidentiality. There are several mechanisms that can be used for this purpose; the most important ones are discussed here. These are:

No authentication

Basic authentication

Simple Authentication and Security Layer (SASL)

Transport Layer Security (TLS)

The security mechanism to be used in LDAP is negotiated when the connection between the client and the server is established.

##### No authentication

No authentication should only be used when data security is not an issue and when no special access control permissions are involved. This might be the case, for example, when your directory is an address book browsable by anybody. No authentication is assumed when you leave the password and DN field empty in the bind API call. The LDAP server then automatically assumes an anonymous user session and grants access with the appropriate access controls defined for this kind of access (not to be confused with the SASL anonymous user discussed in “Simple Authentication and Security Layer (SASL)”).

##### Basic authentication

Basic authentication is also used in several other Web-related protocols, such as HTTP. When using basic authentication with LDAP, the client identifies itself to the server by means of a DN and a password, which are sent in the clear over the network (some implementations might use Base64 encoding instead). The server considers the client authenticated if the DN and password sent by the client matches the password for that DN stored in the directory. Base64 encoding is defined in the Multipurpose Internet Mail Extensions, or MIME (see 15.3, “Multipurpose Internet Mail Extensions (MIME)” on page 571). Base64 is a relatively simple encryption, and it is not hard to break after the data has been captured in the network.

##### Simple Authentication and Security Layer (SASL)

SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols, and is defined in RFC 4422. It has been added to LDAPv3 to overcome the authentication shortcomings of Version 2. SASL was originally devised to add stronger authentication to the IMAP protocol (see 15.5, “Internet Message Access Protocol (IMAP4)” on page 591), but has since evolved into a more general system for mediating between protocols and authentication systems.

In SASL, connection protocols, such as LDAP, IMAP, and so on, are represented by profiles; each profile is considered a protocol extension that allows the protocol and SASL to work together. A complete list of SASL profiles can be obtained from the Information Sciences Institute (ISI). Among these are IMAP, SMTP, POP, and LDAP. Each protocol that intends to use SASL needs to be extended with a command to identify an authentication mechanism and to carry out an authentication exchange. Optionally, a security layer can be negotiated to encrypt the data after authentication and ensure confidentiality. LDAPv3 includes such a command (**ldap\_sasl\_bind()** or **ldap\_sasl\_bind\_s()**). The key parameters that influence the security method used are:

|  |  |
| --- | --- |
| **dn** | This is the distinguished name of the entry which is to bind. This can be thought of as the user ID in a normal user ID and password authentication. |
| **mechanism** | This is the name of the security method to use. Valid security mechanisms are, currently:  **OTP** The One Time Password mechanism  (defined in RFC 2444).  **GSSAPI** The Generic Security Services Application Program Interface (defined in RFC 2743).  **CRAM-MD5** The Challenge/Response Authentication Mechanism (defined in RFC 2195), based on the HMAC-MD5 MAC algorithm.  **DIGEST-MD5** An HTTP Digest-compatible CRAM based on the HMAC -MD5 MAC algorithm.  **EXTERNAL** An external mechanism. Usually this is TLS, discussed in “Transport Layer Security (TLS)” on page 474.  **ANONYMOUS** Unauthenticated access. |
| **credentials** | This contains the arbitrary data that identifies the DN. The format and content of the parameter depends on the mechanism chosen. If it is, for example, the ANONYMOUS  mechanism, it can be an arbitrary string or an e-mail address that identifies the user. |

SASL provides a high-level framework that lets the involved parties decide on the particular security mechanism to use. The SASL security mechanism negotiation between client and server is done in the clear. After the client and the server agree on a common mechanism, the connection is secure against modifying the authentication identities. However, an attacker might try to eavesdrop the mechanism negotiation and cause a party to use the least secure mechanism. In order to prevent this from happening, configure clients and servers to use a minimum security mechanism, provided they support such a configuration option. As stated earlier, SSL and its successor, TLS, are the mechanisms commonly used in SASL for LDAP. For details about these protocols, refer to 22.7, “Secure Sockets Layer (SSL)” on page 854.

Because no data encryption method was specified in LDAPv2, some vendors added their own SSL calls to the LDAP API. A potential drawback of such an approach is that the API calls might not be compatible among different vendor implementations. The use of SASL, as specified in LDAPv3, assures compatibility, although it is likely that vendor products will support only a subset of the possible range of mechanisms (possibly only SSL).

##### Transport Layer Security (TLS)

Transport Layer Security (TLS) is available through the SASL EXTERNAL method, described earlier. An LDAP client can opt to secure a session using TLS at any point during a transaction with an LDAP server, except when:

The session is already TLS protected.

A multi-stage SASL negotiation is in progress.

The client is awaiting a response from an operation request.

To request that TLS be set up on the session, the client sends a *StartTLS* message to the server. This enables the client and server to exchange certificates. RFC 4513 requires that, in addition to this, the client verifies the server’s identity using the DNS or IP address presented in the server’s certificate. This prevents a client’s attempt to connect to a server from being intercepted by malicious user, who might then stage a *man-in-the-middle* attack. After this has occurred, the client and server can negotiate a ciphersuite.

#### 12.4.6 LDAP URLs

Because LDAP has become an important protocol on the Internet, a URL format for LDAP resources has been defined in RFC 4516. LDAP URLs begin with ldap:// or ldaps://, if the LDAP server communicates using SSL. LDAP URLs can simply name an LDAP server, or can specify a complex directory search.

Some examples help make the format of LDAP URLs clear. The following example refers to the LDAP server on the host ldpserv.mydiv.mycorp.com (using the well-known port 389):

ldap://ldpserv.mydiv.mycorp.com/

Additionally, search options can be specified in the URL. The following example retrieves all the attributes for the DN ou=Accounting,c=US from the LDAP server on host ldpserv.mydiv.mycorp.com. In this case, nonstandard port 4389 is explicitly specified here as an example. ldap://ldpserv.mydiv.mycorp.com:4389/ou=Accounting,c=US

The following example retrieves all the attributes for the DN cn=JohnSmith,ou=Sales,o=myCorp,c=US. Note that some characters are considered unsafe in URLs because they can be removed or treated as delimiters by some programs. Unsafe characters such as space, comma, brackets, and so forth need to be represented by their hexadecimal value preceded by the percent sign: ldap://ldpserv.mydiv.mycorp.com/cn=John%20Smith,o=myCorp,c=US

In this example, %20 is a space. More information about unsafe characters and URLs in general are in RFC 4516.

In addition to options, the URL can specify what values attributes are to be returned using the ? symbol. For example, assume we want to find the U.S. address of myCorp. We use the following URL: ldap://ldpserv.mydiv.mycorp.com:4389/o=myCorp,c=US?postalAddress

#### 12.4.7 LDAP and DCE

DCE has its own Cell Directory Service, or CDS (see 13.3.1, “DCE directory service” on page 498). If applications never access resources outside of their DCE cell, only CDS is required. However, if an application needs to communicate with resources in other DCE cells, the Global Directory Agent (GDA) is required. The GDA accesses a global (that is, non-CDS) directory where the names of

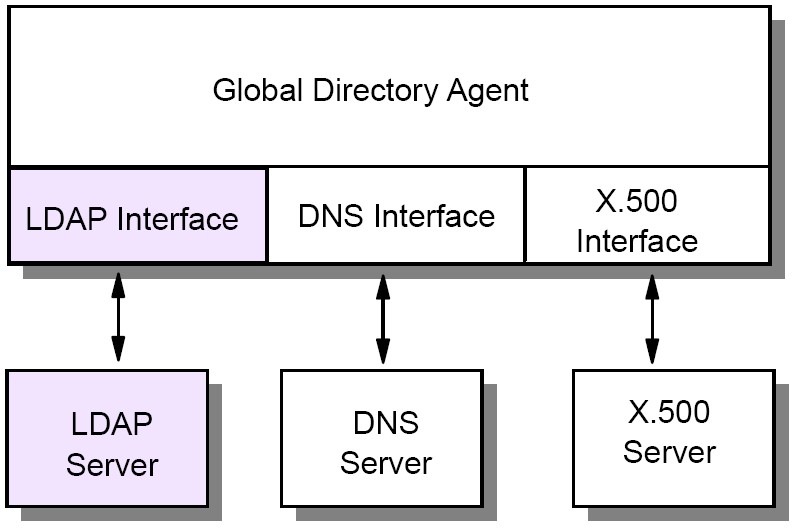
DCE cells can be registered. This global directory (GDS) can be either a Domain Name System (DNS) directory or an X.500 directory. The GDA retrieves the address of a CDS server in the remote cell. The remote CDS can then be contacted to find DCE resources in that cell. Using the GDA enables an organization to link multiple DCE cells together using either a private directory on an intranet or a public directory on the Internet.

In view of LDAP's strong presence in the Internet, two LDAP projects have been sponsored by The Open Group to investigate LDAP integration with DCE technology.

##### LDAP interface for the GDA

One way LDAP is being integrated into DCE is to allow DCE cells to be registered in LDAP directories. The GDA in a cell that wants to connect to remote cells is configured to enable access to the LDAP directory (see Figure 12-17).

*Figure 12-17 The LDAP interface for GDA*



DCE originally only supported X.500 and DNS name syntax for cell names. LDAP and X.500 names both follow the same hierarchical naming model, but their syntax is slightly different. X.500 names are written in reverse order and use a slash (/) rather than a comma (,) to separated relative distinguished names. When the GDA is configured to use LDAP, it converts cell names in X.500 format into the LDAP format.

##### LDAP interface for the CDS

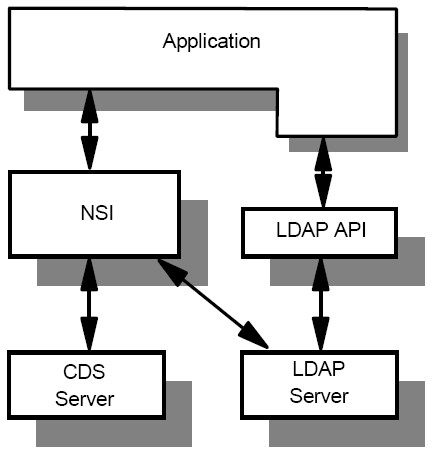
DCE provides two programming interfaces to the Directory Service; Name

Service Interface (NSI) and the X/Open Directory Service (XDS). XDS is an X.500-compatible interface used to access information in the GDS, and it can also be used to access information in the CDS. However, the use of NSI is much more common in DCE applications. The NSI API provides functionality that is

specifically tailored for use with DCE client and server programs that use RPC. NSI allows servers to register their address and the type of RPC interface they support. This address/interface information is called an RPC binding, and is needed by clients that want to contact the server. NSI allows clients to search the CDS for RPC binding information.

NSI was designed to be independent of the directory where the RPC bindings are stored. However, the only supported directory to date has been CDS. NSI will be extended to also support adding and retrieving RPC bindings from an LDAP directory. This will allow servers to advertise their RPC binding information in either CDS or an LDAP directory. Application programs can use either the NSI or the LDAP API when an LDAP directory is used (see Figure 12-18).

*Figure 12-18 The LDAP interface for NSI*



#### 12.4.8 The Directory-Enabled Networks (DEN) initiative

In September 1997, Cisco Systems Inc. and Microsoft® Corp. announced the so-called Directory-Enabled Networks (DEN) initiative as a result of a collaborative work. Many companies, such as IBM, either support this initiative or

actively participate in ad hoc working groups (ADWGs). DEN represents an information model specification for an integrated directory that stores information about people, network devices, and applications. The DEN schema defines the object classes and their related attributes for those objects. As such, DEN is a key piece to building intelligent networks, where products from multiple vendors can store and retrieve topology and configuration-related data.

Of special interest is that the DEN specification defines LDAPv3 as the core protocol for accessing DEN information, which makes information available to LDAP-enabled clients or network devices, or both.

DEN makes use of the Common information Model (CIM). CIM details a way of integrating different management models such as SNMP MIBs and DMTF MIFs. At the time of writing, the most current CIM schema was version 2.12, released in April of 2006.

More information about the DEN initiative can be found on the founder’s Web at:

<http://www.dmtf.org/standards/wbem/den/><http://www.dmtf.org/standards/cim/>

#### 12.4.9 Web-Based Enterprise Management (WBEM)

WBEM is a set of standards designed to deliver an integrated set of management tools for the enterprise. By making use of XML and CIM, it becomes possible to manage network devices, desktop systems, telecom systems and application systems, all from a Web browser. For further information, see: <http://www.dmtf.org/standards/wbem/>

### 12.5 RFCs relevant to this chapter

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

[RFC 1032 – Domain administrators guide (November 1987)](http://www.ietf.org/rfc/rfc1032.txt)

[RFC 1033 – Domain administrators operations guide (November 1987)](http://www.ietf.org/rfc/rfc1033.txt)

[RFC 1034 – Domain names - concepts and facilities (November 1987)](http://www.ietf.org/rfc/rfc1034.txt)

[RFC 1035 – Domain names - implementation and specifications](http://www.ietf.org/rfc/rfc1035.txt)

[(November](http://www.ietf.org/rfc/rfc1035.txt) [1987)](http://www.ietf.org/rfc/rfc1035.txt)

[RFC 1101 – DNS encoding of network names and other types (April 1989)](http://www.ietf.org/rfc/rfc1101.txt)

[RFC 1183 – New DNS RR Definitions (October 1990)](http://www.ietf.org/rfc/rfc1183.txt)

[RFC 1202 – Directory Assistance service (February 1991)](http://www.ietf.org/rfc/rfc1202.txt)

[RFC 1249 – DIXIE Protocol Specification (August 1991)](http://www.ietf.org/rfc/rfc1249.txt)

[RFC 1348 – DNS NSAP RRs (July 1992)](http://www.ietf.org/rfc/rfc1348.txt)

[RFC 1480 – The US Domain (June 1993)](http://www.ietf.org/rfc/rfc1480.txt)

[RFC 1706 – DNS NSAP Resource Records (October 1994)](http://www.ietf.org/rfc/rfc1706.txt)

[RFC 1823 – The LDAP Application Programming Interface (August 1995)](http://www.ietf.org/rfc/rfc1823.txt)

[RFC 1876 – A Means for Expressing Location Information in the Domain](http://www.ietf.org/rfc/rfc1876.txt)

[Name System (January 1996)](http://www.ietf.org/rfc/rfc1876.txt)

[RFC 1995 – Incremental Zone Transfer in DNS (August 1996)](http://www.ietf.org/rfc/rfc1995.txt)

[RFC 1996 – A Mechanism for Prompt Notification of Zone Changes (DNS](http://www.ietf.org/rfc/rfc1996.txt)

[NOTIFY) (August 1996)](http://www.ietf.org/rfc/rfc1996.txt)

[RFC 2136 – Dynamic Updates in the Domain Name System (DNS UPDATE) (April 1997)](http://www.ietf.org/rfc/rfc2136.txt)

[RFC 2444 – The One-time-Password SASL Mechanism (October 1998)](http://www.ietf.org/rfc/rfc2444.txt)

RFC 4592 – The Role of Wildcards in the Domain Name System (July 2006)

[RFC 2743 – Generic Security Service Application Program Interface Version](http://www.ietf.org/rfc/rfc2743.txt)

[2, Update 1 (January 2000)](http://www.ietf.org/rfc/rfc2743.txt)

[RFC 2874 – DNS Extensions to Support IPv6 Address Aggregation and](http://www.ietf.org/rfc/rfc2874.txt)

[Renumbering (July 2000)](http://www.ietf.org/rfc/rfc2874.txt)

[RFC 3007 – Secure Domain Name Systems (DNS) Dynamic Update](http://www.ietf.org/rfc/rfc3007.txt)

[(November 2000)](http://www.ietf.org/rfc/rfc3007.txt)

[RFC 3494 – Lightweight Directory Access protocol version 2 (LDAPv2) (March 2003)](http://www.ietf.org/rfc/rfc3494.txt)

[RFC 3596 – DNS Extensions to Support IP Version 6 (October 2003)](http://www.ietf.org/rfc/rfc3596.txt)

[RFC 3645 – Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG) (October 2003)](http://www.ietf.org/rfc/rfc3645.txt)

[RFC 3901 – DNS IPv6 Transport Operational Guidelines (September 2004)](http://www.ietf.org/rfc/rfc3901.txt)

[RFC 4033 – DNS Security Introduction and Requirements (March 2005)](http://www.ietf.org/rfc/rfc4033.txt)

[RFC 4034 – Resource Records for the DNS Security Extensions](http://www.ietf.org/rfc/rfc4034.txt)

[(March](http://www.ietf.org/rfc/rfc4034.txt) [2005)](http://www.ietf.org/rfc/rfc4034.txt)

[RFC 4035 – Protocol Modifications for the DNS Security Extensions](http://www.ietf.org/rfc/rfc4035.txt)

[(March](http://www.ietf.org/rfc/rfc4035.txt) [2005)](http://www.ietf.org/rfc/rfc4035.txt)

[RFC 4339 – IPv6 Host Configuration of DNS Server Information Approaches (February 2006)](http://www.ietf.org/rfc/rfc4339.txt)

[RFC 4398 – Storing Certificates in the Domain Name System (DNS)](http://www.ietf.org/rfc/rfc4398.txt)

[(March](http://www.ietf.org/rfc/rfc4398.txt) [2006)](http://www.ietf.org/rfc/rfc4398.txt)

[RFC 4422 – Simple Authentication and Security Layer (SASL) (June 2006)](http://www.ietf.org/rfc/rfc4422.txt)

[RFC 4501 – Domain Name System Uniform Resource Identifiers (May 2006)](http://www.ietf.org/rfc/rfc4501.txt) [RFC 4505 – Anonymous Simple Authentication and Security Layer (SASL) (June 2006)](http://www.ietf.org/rfc/rfc4505.txt)

[RFC 4510 – Lightweight Directory Access Protocol (LDAP): Technical](http://www.ietf.org/rfc/rfc4510.txt)

[Specification Road Map (June 2006)](http://www.ietf.org/rfc/rfc4510.txt)

[RFC 4511 – Lightweight Directory Access Protocol (LDAP): The Protocol (June 2006)](http://www.ietf.org/rfc/rfc4511.txt)

[RFC 4512 – Lightweight Directory Access Protocol (LDAP): Directory](http://www.ietf.org/rfc/rfc4512.txt)

[Information Models (June 2006)](http://www.ietf.org/rfc/rfc4512.txt)

[RFC 4513 – Lightweight Directory Access Protocol (LDAP): Authentication](http://www.ietf.org/rfc/rfc4513.txt)

[Methods and Security Mechanisms (June 2006)](http://www.ietf.org/rfc/rfc4513.txt)

[RFC 4514 – Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names (June 2006)](http://www.ietf.org/rfc/rfc4514.txt)

[RFC 4515 – Lightweight Directory Access Protocol (LDAP): String](http://www.ietf.org/rfc/rfc4515.txt)

[Representation of Search Filters (June 2006)](http://www.ietf.org/rfc/rfc4515.txt)

[RFC 4516 – Lightweight Directory Access Protocol (LDAP): Uniform](http://www.ietf.org/rfc/rfc4516.txt)

[Resource Locator (June 2006)](http://www.ietf.org/rfc/rfc4516.txt)

[RFC 4517 – Lightweight Directory Access Protocol (LDAP): Syntaxes and](http://www.ietf.org/rfc/rfc4517.txt)

[Matching Rules (June 2006)](http://www.ietf.org/rfc/rfc4517.txt)

[RFC 4518 – Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation (June 2006)](http://www.ietf.org/rfc/rfc4518.txt)

[RFC 4519 – Lightweight Directory Access Protocol (LDAP): Schema for User Applications (June 2006)](http://www.ietf.org/rfc/rfc4519.txt)

[RFC 4520 – Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4520.txt)

[RFC 4521 – Considerations for Lightweight Directory Access Protocol](http://www.ietf.org/rfc/rfc4521.txt)

[(LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4521.txt)

[RFC 4522 – Lightweight Directory Access Protocol (LDAP): The Binary](http://www.ietf.org/rfc/rfc4522.txt)

[Encoding Option (June 2006)](http://www.ietf.org/rfc/rfc4522.txt)

[RFC 4523 – Lightweight Directory Access Protocol (LDAP): Schema](http://www.ietf.org/rfc/rfc4523.txt)

[Definitions for X.509 Certificates (June 2006)](http://www.ietf.org/rfc/rfc4523.txt)

[RFC 4524 – Lightweight Directory Access Protocol (LDAP): COSINE/LDAP X.500 Schema (June 2006)](http://www.ietf.org/rfc/rfc4524.txt)

[RFC 4525 – Lightweight Directory Access Protocol (LDAP): Modify-Increment](http://www.ietf.org/rfc/rfc4525.txt)

[Extension (June 2006)](http://www.ietf.org/rfc/rfc4525.txt)

[RFC 4526 – Lightweight Directory Access Protocol (LDAP): Absolute True and False Filters (June 2006)](http://www.ietf.org/rfc/rfc4526.txt)

[RFC 4527 – Lightweight Directory Access Protocol (LDAP): Read Entry Controls (June 2006)](http://www.ietf.org/rfc/rfc4527.txt)

[RFC 4528 – Lightweight Directory Access Protocol (LDAP): Assertion Control (June 2006)](http://www.ietf.org/rfc/rfc4528.txt)

[RFC 4529 – Requesting Attributes by Object Class in the Lightweight](http://www.ietf.org/rfc/rfc4529.txt)

[Directory Access Protocol (LDAP) (June 2006)](http://www.ietf.org/rfc/rfc4529.txt)

[RFC 4530 – Lightweight Directory Access Protocol (LDAP): entryUUID](http://www.ietf.org/rfc/rfc4530.txt)

[Operational Attribute (June 2006)](http://www.ietf.org/rfc/rfc4530.txt)

[RFC 4531 – Lightweight Directory Access Protocol (LDAP): Turn Operation (June 2006)](http://www.ietf.org/rfc/rfc4531.txt)

[RFC 4532 – Lightweight Directory Access Protocol (LDAP): “Who Am I?”](http://www.ietf.org/rfc/rfc4532.txt)

[Operation (June 2006)](http://www.ietf.org/rfc/rfc4532.txt)

[RFC 4533 – Lightweight Directory Access Protocol (LDAP): Content](http://www.ietf.org/rfc/rfc4533.txt)

[Synchronization Operation (June 2006)](http://www.ietf.org/rfc/rfc4533.txt)

1. Some of these protocols can be described as impractical at best. For instance, RFC 1149 (dated 1990 April 1) describes the transmission of IP datagrams by carrier pigeon and RFC 1437 (dated 1993 April 1) describes the transmission of people by electronic mail. [↑](#footnote-ref-1)
2. RFC 1771 uses uppercase to name BGP messages. The same convention is used in this section. [↑](#footnote-ref-2)
3. At the time of writing, there were 13 root servers. [↑](#footnote-ref-3)
4. Note that all of the fields are byte-aligned. The alignment of the Type field on a 4-byte boundary is for example purposes and is not required by the format. [↑](#footnote-ref-4)