

1. History of TCP/IP

TCP/IP was initially designed to meet the data communication needs of the U.S. Department of Defence (DOD).

In the late 1960s the Advanced Research Projects Agency (ARPA, now called DARPA) of the U.S. Department of Defence began a partnership with U.S. universities and the corporate research community to design open, standard protocols and build multi-vendor networks.

Together, the participants planned ARPANET, the first packet switching network. The first experimental four-node version of ARPANET went into operation in 1969. These four nodes at three different sites were connected together via 56 kbit/s circuits, using the Network Control Protocol (NCP). The experiment was a success, and the trial network ultimately evolved into a useful operational network, the "ARPA Internet".

In 1974, the design for a new set of core protocols, for the ARPANET, was proposed in a paper by Vinton G. Cerf and Robert E. Kahn. The official name for the set of protocols was TCP/IP Internet Protocol Suite, commonly referred to as TCP/IP, which is taken from the names of the network layer protocol (Internet protocol [IP]) and one of the transport layer protocols (Transmission Control Protocol [TCP]).

TCP/IP is a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic.

The initial specification went through four early versions, culminating in version 4 in 1979.

History of TCP/IP

- ◆ 1969: ARPANET went into operation
 - four packet-switched nodes at three different sites
 - connected together via 56 kbit/s circuits
 - using the Network Control Protocol (NCP)
 - funded by the U.S. Department of Defence
- ◆ 1974: TCP/IP designed by Vinton G. Cerf and Robert E. Kahn
- ◆ 1979: IP version 4 documented
- ◆ 1979: the Internet Control and Configuration Board (ICCB) formed
- ◆ 1979: BSD Unix with TCP/IP supplied to Universities
- ◆ 1980: ARPA started converting machines to TCP/IP
- ◆ 1983: mandate that all computers connected to ARPANET use TCP/IP
- ◆ 1983 ARPANET split into two separate networks,
 - ARPANET for further research
 - MILNET for the military

2. History of the Internet

By 1985, the ARPANET was heavily used and congested. In response, the National Science Foundation (NSF) initiated phase one development of the NSFNET. ARPANET was officially decommissioned in 1989. The NSFNET was composed of multiple regional networks and peer networks (such as the NASA Science Network) connected to a major backbone that constituted the core of the overall NSFNET

In its earliest form, in 1986, the NSFNET created a three-tiered network architecture. The architecture connected campuses and research organisations to regional networks, which in turn connected to a main backbone linking six nationally funded super-computer centres. The original links were 56 kbit/s.

The links were upgraded in 1988 to faster T1 (1.544 Mbit/s) links as a result of the NSFNET 1987 competitive solicitation for a faster network service, awarded to Merit Network, Inc. and its partners MCI, IBM, and the state of Michigan. The NSFNET T1 backbone connected a total of 13 sites that included Merit, BARRNET, MIDnet, Westnet, NorthWestNet, SESQUINET, SURANet, NCAR (National Centre of Atmospheric Research), and five NSF supercomputer centres.

In 1991 the NSF decided to move the backbone to a private company and start charging institutions for connections. In 1991, Merit, IBM, and MCI started a not-for-profit company named Advanced Networks and Services (ANS). By 1993, ANS had installed a new network that replaced NFSNET. Called ANSNET, the new backbone operated over T3 (45 Mbit/s) links. ANS owned this new Wide Area Network (WAN), unlike previous WANs used in the Internet, which had all been owned by the U.S. government.

In 1993, NSF invited proposals for projects to accommodate and promote the role of commercial service providers and lay down the structure of a new and robust Internet model. At the same time, NSF withdrew from the actual operation of the network and started to focus on research aspects and initiatives.

The "NSF solicitation" included four separate projects for which proposals were invited:

- Creating a set of Network Access Points (NAPs) where major providers connect their networks and exchange traffic.
- Implementing a Route Arbiter (RA) project, to provide equitable treatment of the various network service providers with regard to routing administration.
- Providing a very high-speed Backbone Network Service (vBNS) for educational and governmental purposes.
- Moving existing "regional" networks, from the NSFNET backbone to other Network Service Providers (NSPs) which have connections to NAPs.

Partly as a result of the NSF solicitations, today's Internet structure has moved from a core network (NSFNET) to a more distributed architecture operated by commercial providers such as Sprint, MCI, BBN, and others connected via major network exchange points, called Network Access Points (NAPs). A NAP is defined as a high-speed switch to which a number of routers can be

connected for the purpose of traffic exchange. This allows Internet traffic from the customers of one provider to reach the customers of another provider. Internet Service Providers (ISPs) are companies that provide Internet services, for example, Web access and Internet mail, to end customers, both individual users and corporate users. The connection point between a customer and an ISP is called a point of presence (POP). The physical connection between a customer and an ISP can be provided by many different physical access methods, for example dial-up or Frame Relay.

ISP networks exchange information with each other by connecting to NSPs that are connected to NAPs, or by connecting directly to NAPs.

The NSFNET was physically connected to the following four NAPS between 1993 and 1995: (1) Sprint NAP, Pennsauken, NJ (2) PacBell NAP, San Francisco, CA (3) Ameritech Advanced Data Services (AADS) NAP, Chicago, IL (4) MFS Datanet (MAE-East) NAP, Washington, D.C. Additional NAPs continue to be created around the world as providers keep finding the need to interconnect.

In 1995 the NSF awarded MCI the contract to build the very high performance Backbone Network Service (vBNS) to replace ANSNET.

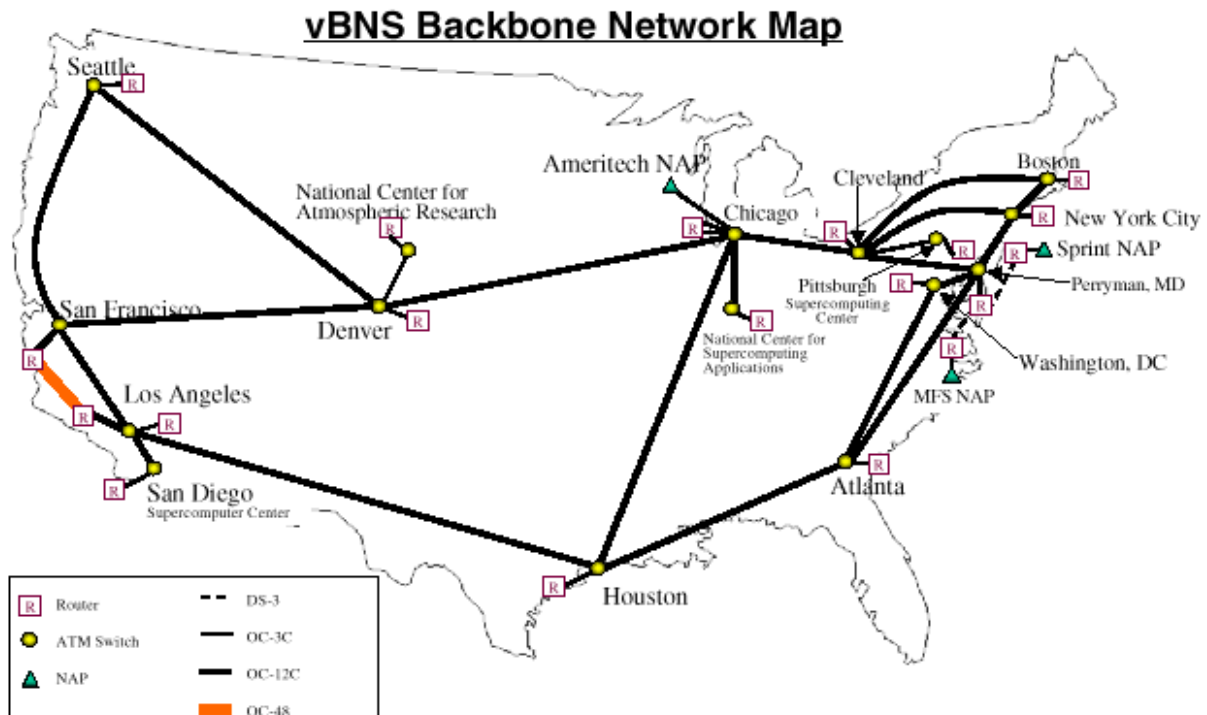
The vBNS was designed for the scientific and research communities. It originally provided high speed interconnection among NSF supercomputing centres and connection to NSF-specified Network Access Points. Today (1999) the vBNS connects two NSF supercomputing centres and research institutions that are selected under the NSF's high performance connections program. MCI owns and operates the network, but cannot determine who connects to it.

The NSF awards grants under its high performance connection program. The vBNS is only available for research projects with high-bandwidth uses and is not used for general Internet traffic.

The vBNS is a US based network that operates at a speed of 622 megabits per second (OC12) using MCI's network of advanced switching and fibre optic transmission technologies.

The vBNS relies on advanced switching and optical fibre transmission technologies, known as Asynchronous Transfer Mode (ATM) and Synchronous Optical Network (SONET). The combination of ATM and SONET enables very high speed, high capacity voice, data, and video signals to be combined and transmitted "on demand".

The vBNS's speeds are achieved by connecting Internet Protocol (IP) through an ATM switching matrix, and running this combination on the SONET network.



3. Internet Architecture Board (IAB)

The IAB, with responsibility for the Internet architecture, was reorganised in 1989 to include a wider community. The new IAB organisation consisted of: (1) an elected IAB chairman and IAB members, (2) the Internet Research Task Force (IRTF), (3) the Internet Engineering Task Force (IETF) and (4) the Internet Assigned Numbers Authority (IANA). The structure is illustrated in the diagram.

The IETF has engineering working groups, the IRTF has research groups, and each has a steering group. The IANA, chartered by the IAB, assigned or co-ordinated all numbers associated with the TCP/IP protocol suite and the Internet. The IETF, IRTF, and IANA remain active today.

The IAB was reorganised in 1992 when it was brought under the auspices of the Internet Society (ISOC), an international body. The IAB was renamed the Internet Architecture Board, but the functions remain reasonably unchanged.

The ISOC is governed by a board of 14 trustees (including five officers) and an executive director. The officers include the president, two vice presidents, a secretary, and a treasurer. The board of trustees has an advisory council and a secretariat.

The individual members of the ISOC elect trustees for three-year terms. Volunteers manage the infrastructure of the ISOC, including members of the IAB and its task forces. Although several government agencies continue to support key aspects of the TCP/IP protocol development, the majority of personal activity (for example, attending meetings writing RFCs) is done on a voluntary basis.

The IAB is the co-ordinating committee for Internet design, engineering and management. The IAB has a maximum of 15 members who work on a

voluntary basis. Individuals are nominated for membership to the IAB by Internet community members and selected by the ISOC trustees for two-year, renewable terms. The IAB creates task forces, committees, and working groups as required within the scope of the IAB's responsibility.

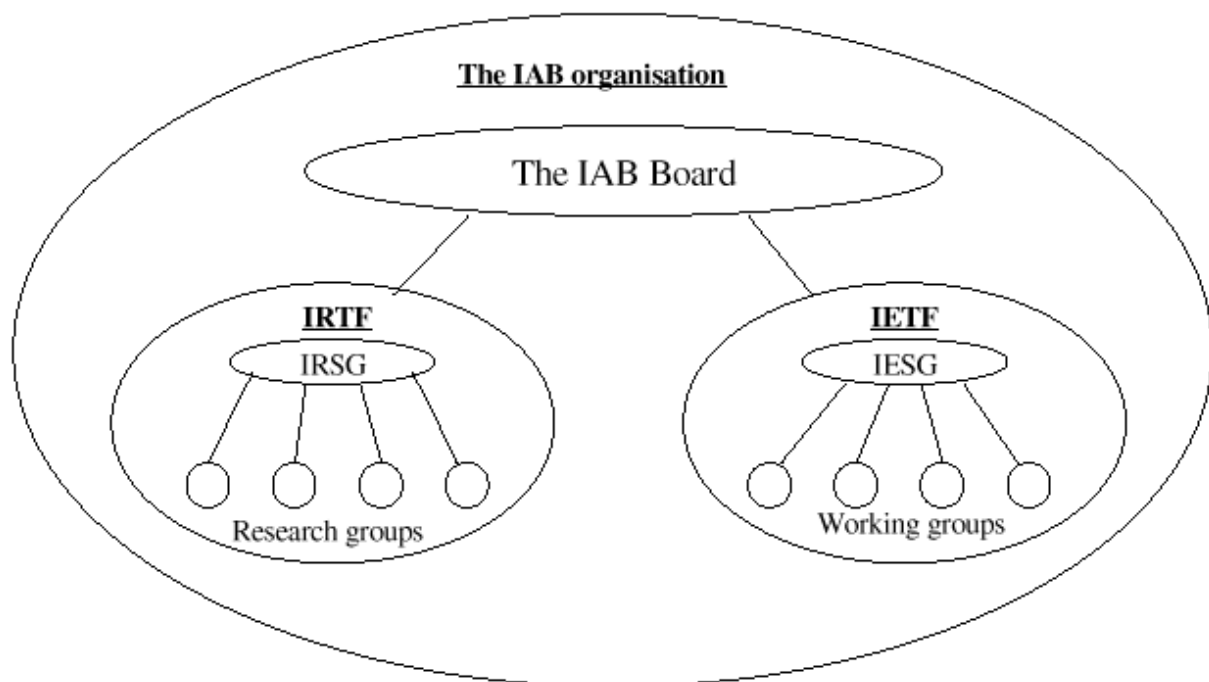
The initial appointments are the following: the editor of the RFC publication series and the chairs of the IETF and the IRTF.

Members of the IAB appoint the chair of the IAB who then has the authority to organise and direct task forces as deemed necessary.

The Internet Engineering Steering Group (IESG) members are nominated by the Internet community and selected by the IAB. All terms are two years renewable. The chairman and the IESG members organise and manage the IETF.

There is an overlap of functions and membership between the IETF and the IRTF, with the major difference being viewpoint and sometimes time frame. This overlap is deliberate and considered vital for technology transfer. The following sections briefly describe the IETF, IRTF and IANA.

Internet Architecture Board (IAB) Organisation



4. Internet Engineering Task Force (IETF)

The IETF co-ordinates the operation, management and evolution of the Internet protocols. The IETF does not deal with the operation of any Internet network, nor does it set any operational policies. Its charter is to specify the protocols and architecture of the Internet and recommend standards for IAB approval.

The IETF is organised in relation to several technical areas. These areas, which change periodically, would typically include the 8 areas listed in the

diagram. Details on each of these groups can be obtained from the IETF home page (www.ietf.org)

The IETF chairperson and a technical area director from each area make up the IESG membership. Each technical area director has a responsibility for a subset of all IETF working groups. There are many working groups, each with a narrow focus and the goal of completing a specific task before moving onto a new task.

The IETF is the major source of proposed protocol standards for final approval by the IESG. The IETF meets three times annually, and extensive minutes as well as reports from each of the working groups are issued by the IETF secretariat.

Active IETF Working Groups

- ◆ Applications
- ◆ Internet
- ◆ Operations and Management
- ◆ Routing
- ◆ Security
- ◆ Transport
- ◆ User services
- ◆ General

4.1 Internet Research Task Force (IRTF)

The IRTF is a community of network researchers that make up the eight IRTF work groups, listed in the diagram. Details on each of these groups can be obtained from the IRTF home page (www.irtf.org)

The IRTF is concerned with understanding technologies and how they may be used in the Internet, rather than products or standard protocols. However, specific experimental protocols may be developed, implemented and tested to gain the required understanding.

The work of the IRTF is governed by its IRSG. The chairman of the IRTF and the IRSG appoint a chair for each research group (RG). Each RG typically has 10 to 20 members and covers a broad area of research, which is determined by the interests of the members and by recommendations from the IAB.

Internet Research Task Force

● Active IRTF Research Groups

- End-to-End
- Information Infrastructure Architecture
- Internet Resource Discovery
- Network Management
- Reliable Multicast
- Routing
- Secure Multicast
- Services Management

4.2 Internet Assigned Number Authority (IANA)

The Internet employs a central Internet Assigned Numbers Authority (IANA) for the allocation and assignment of various numeric identifiers needed for the operation of the Internet. The IANA function is performed by the University of Southern California's Information Sciences Institute. The IANA is chartered by the IAB to co-ordinate the assigned values of protocol parameters, including type codes, protocol numbers, port numbers, Internet addresses, and Ethernet addresses.

The IANA delegates the responsibility of assigning IP network numbers and domain names to three Regional Internet Registries (RIRs):

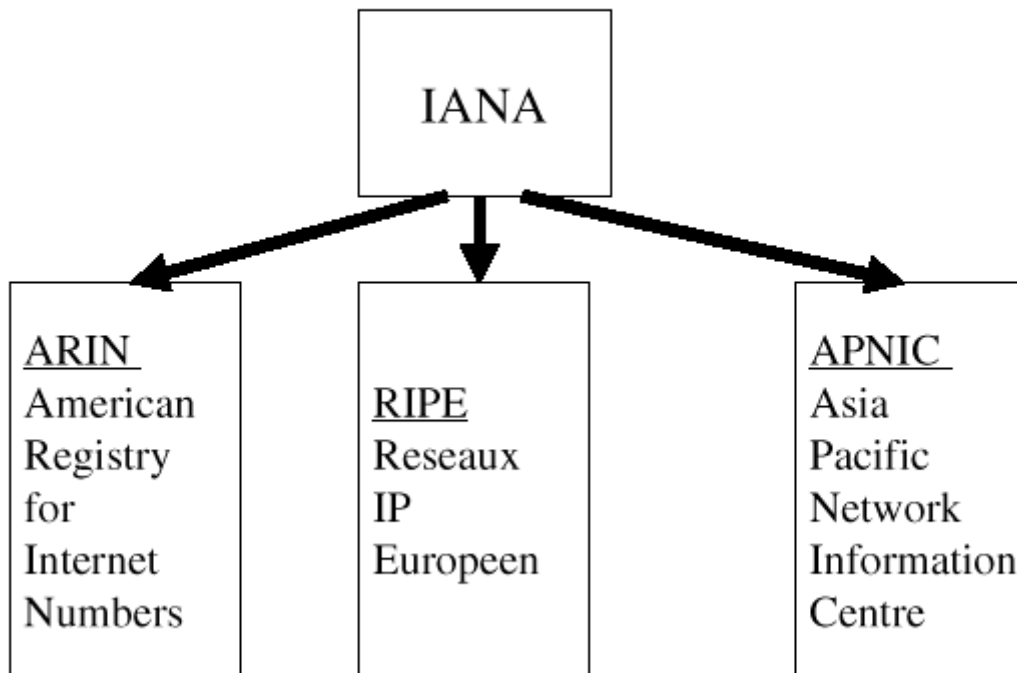
- ARIN (American Registry for Internet Numbers)
- RIPE (Reseaux IP European)
- APNIC (Asia Pacific Network Information Centre)

The registries provide databases and information servers such as WHOIS registry for domains, networks, AS numbers, and their associated Point Of Contacts (POCs).

The documents distributed by the Internet registries include network information, and procedures, including application forms, to request network numbers and register domain name servers. All of these are available from the relevant web sites: www.arin.org, www.ripe.org, www.apnic.org.

The RIPE web site contains a list that shows all (ISO 3166 defined) countries listed in the three RIR areas, (www.ripe.net/info/ncc/rir-areas.html). For the RIPE area there are also geographical maps which show the RIPE area countries and which list the Local Internet Registries (LIRs) in each country.

Internet Assigned Number Authority (IANA)



4.3 Request for Comments (RFCs)

Documentation of work on the Internet, proposals for new or revised protocols, and TCP/IP protocol standards all appear in a series of technical reports called Internet Request for Comments, or RFCs. Preliminary versions of RFCs are known as Internet drafts. RFCs can be short or long, can cover broad concepts or details, and can be standards or merely proposals for new protocols. The RFC editor is a member of the IAB.

The RFC series is numbered sequentially in the chronological order RFCs are written. Each new or revised RFC is assigned a new number, so readers must be careful to obtain the highest numbered version of a document.

Copies of RFCs are available from many sources including the IETF web page (www.ietf.org/rfc.html).

A unique standard (STD) number is assigned to each protocol reaching the maturity level of standard. The STD number identifies one or more RFCs that provide a specification for the protocol. Although the RFC identified by the STD number may change, the STD number is constant.

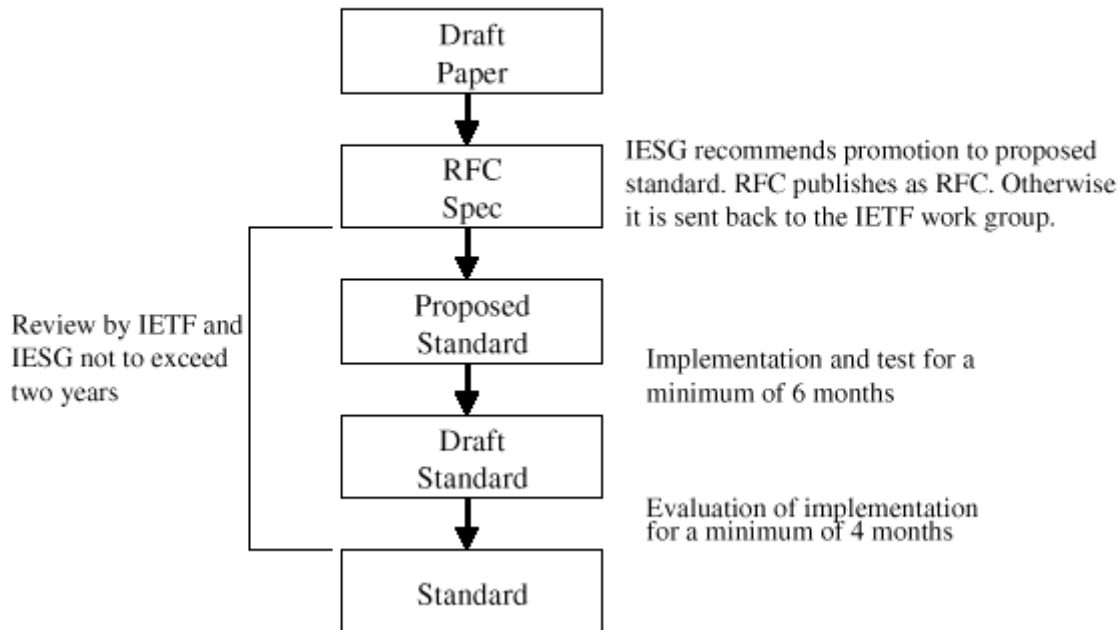
When a new RFC replaces an existing RFC, the existing RFC becomes obsolete. The replaced RFC number (or numbers) are listed under the title of "obsoletes" on the front page of the new RFC.

Standards Track

Each RFC providing a specification of a protocol is assigned a "maturity level" (state of standardisation) and a "requirement level" (status). The maturity level of a Standards Track protocol begins with "proposed" standard. There are also protocols with a maturity level of "experimental". Experimental protocols may

remain experimental indefinitely, become "historic" or be reassigned as a "proposed standard" and enter the standards track. Protocols with a proposed standard maturity level must be implemented and reviewed for a minimum of six months before progressing to "draft standard". Progressing from draft standard to standard requires evaluation for a minimum of four months and the approval of the IESG and the IAB. The diagram illustrates the Standards Track process.

RFC Standards Track Process



Important RFCs

All RFCs can be obtained from the following web address which also contains useful search tools, (www.rfc-editor.org/rfc.html).

- The "Internet Official Protocol Standards", (STD number 1), currently RFC 2500. This RFC describes the state of standardisation of all protocols used in the Internet as determined by the IETF. Each protocol has one of the following states of standardisation: standard, draft standard, proposed standard, experimental, informational or historic. Additionally each protocol has a requirement level: required, recommended, elective, limited use or not recommended.
- The "Assigned Numbers" RFC, (STD number 2), currently RFC 1700, specifies all the numbers and keywords that are used in the Internet protocol.
- The "Requirements for Internet hosts" (STD number 3) RFCs 1122 and 1123. RFC 1122 handles the link layer, network layer, and transport layer, while RFC 1123 handles the application layer. These two RFCs make numerous corrections and interpretations of the important earlier RFCs, and are often the starting point when looking at any of the finer details of a given protocol.
- The "Requirements for IP Version 4 Routers", (proposed standard to update STD number 4, Requirements for Internet Gateways) RFC 1812. This RFC defines and discusses requirements for devices that perform the network layer forwarding function of the Internet protocol suite.

Important RFCs

STD Number	RFC Number	Name
1	2500	Internet Official Protocol Standards
2	1700	Assigned Numbers
3	1122 1123	Requirements for Internet hosts
4*	1812	Requirements for IP Version 4 Routers

www.rfc-editor.org/rfc.html

5. OSI 7-Layer Model

The Physical Layer defines the type of medium, the transmission method, and the transmission rates available for the network.

The Data Link Layer defines how the network medium is accessed: which protocols are used, the packet/framing methods, and the virtual circuit/connection services.

The Network Layer standardises the way in which addressing is accomplished between linked networks.

The Transport Layer handles the task of reliable message delivery and flow control between applications on different devices.

The Session Layer establishes two-way communication between applications running on different devices on the network.

The Presentation layer translates data formats so that devices with different "languages" can communicate

The Application Layer interfaces directly with the application programs running on the devices. It provides services such as file access and transfer, peer-to-peer communication among applications, and resource sharing.

OSI 7- Layer Model

APPLICATION	Interfaces directly with application programs running on the devices.
PRESENTATION	Provides code conversion and data reformatting.
SESSION	Co-ordinates interaction between end-to-end application processes.
TRANSPORT	Provides end-to-end data integrity and quality of service.
NETWORK	Switches and routes information to the appropriate network device.
DATA LINK	Transfers units of information to the other end of the physical link.
PHYSICAL	Transmits/Receives on the network medium

5.1 OSI 7-Layer Model (Contd)

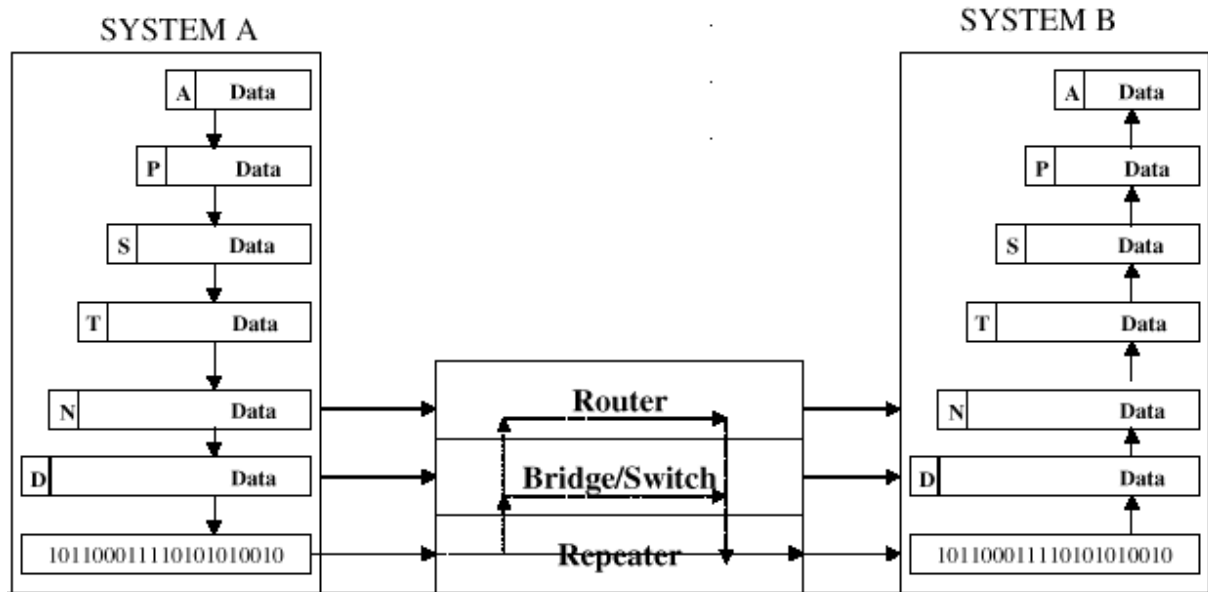
Each layer operates independently of the others using a method referred to as encapsulation. At the sending device, each layer receiving data from the layer above processes the data, adds its own protocol header and transfer the data block to the layer below. The layer below simply treats the data as a data block; it does not try to understand its meaning. The block is processed by the layer, which adds its own protocol header and then passes the larger data block to the layer below. At the receiving device the reverse happens. When the data arrives, the first layer processes its peer header and then passes the data to the layer above which carries out the same action. Ultimately, the application data originally sent by the sending device arrives at the receiving application.

Routers operate at the network layer. They connect networks into internetworks that are physically unified, but in which each network retains its identity as a separate network environment.

Bridges operate at the Data link layer. They connect network environments into logical and physical single internetworks.

Repeaters operate at the Physical layer. They receive transmissions (bits) on a LAN segment and regenerate the bits to boost a degraded signal and extend the length of the LAN segment.

OSI 7- Layer Model and Internetworking Devices



6. TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) is not a single protocol; it refers to a family or suite of protocols. The suite consists of a four-layer model.

Network Interface Layer

The Network Interface Layer is equivalent to the combination of the Physical and Data Link Layers in the OSI model. It is responsible for formatting packets and placing them onto the underlying network. All common Data Link protocols support TCP/IP.

Internet Layer

The Internet Layer is equivalent to the Network Layer in the OSI model. It is responsible for network addressing. The main protocols at this layer are: Internet Protocol (IP), Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), Internet Control Message Protocol (ICMP), and Internet Group Management Protocol (IGMP).

The Transport Layer

The Transport Layer is equivalent to the Transport Layer in the OSI model. The Internet Transport layer is implemented by TCP and the User Datagram Protocol (UDP). TCP provides reliable data transport, while UDP provides unreliable data transport.

The Application Layer

The Application Layer is equivalent to the top three layers, (Application, Presentation and Session Layers), in the OSI model. The Application Layer is responsible for interfacing between user applications and the Transport Layer. Applications commonly used are File Transfer Protocol (FTP), Telnet, Simple Network Management Protocol (SNMP), Domain Name system (DNS), Simple Mail Transfer Protocol (SMTP), and so on.

Internet Protocol Suite and OSI Reference Model

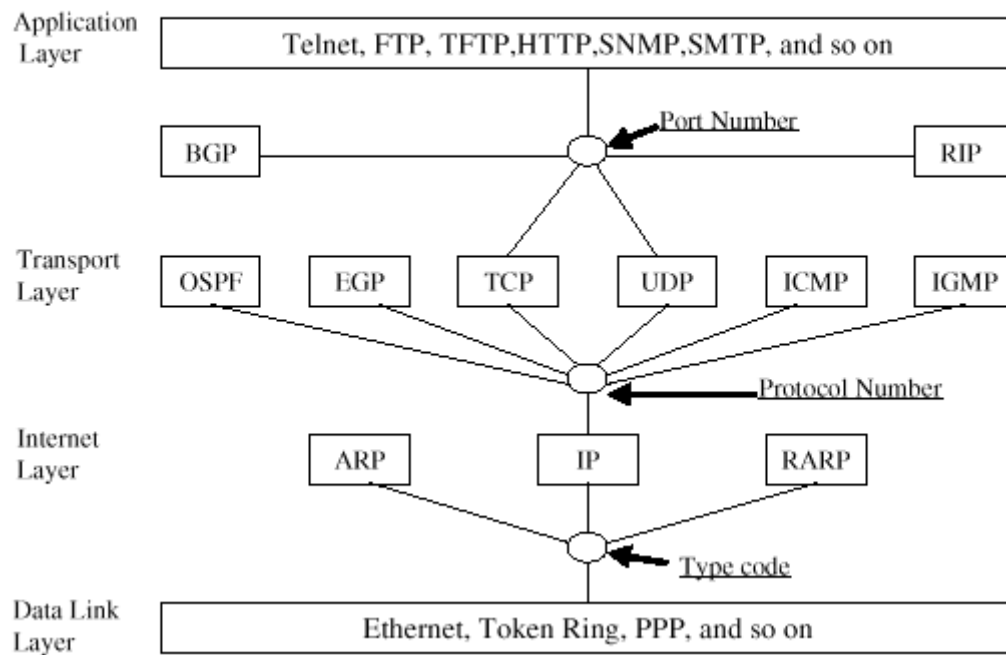
APPLICATION		APPLICATION (FTP, TELNET, SNMP, DNS, SMTP)	
PRESENTATION			
SESSION			
TRANSPORT		TRANSPORT (TCP or UDP)	
NETWORK		ICMP, IGMP	INTERNET PROTOCOL (IP)
DATA LINK		NETWORK INTERFACE (LAN - ETH, TR, FDDI) (WAN - Serial lines, FR, ATM)	
PHYSICAL			

In reality, the interaction between the various protocols is more complex than illustrated in the previous diagram. For example, the Internet Control Message Protocol (ICMP) and the Internet Group Message Protocol (IGMP) are an integral part of the Internet layer. However, each receives data and control in the same manner as a Transport layer function, namely, by an assigned protocol number contained in the IP header. Hence, they are illustrated in this diagram of the TCP/IP protocol stack based on data flow and control. For the same reason, some other protocols may be identified in Internet literature differently than in this illustration. For example, the Routing Information Protocol (RIP) has an assigned port number, contained in a User Datagram Protocol (UDP) header, making it an upper layer protocol. Yet, another routing protocol, Open Shortest Path First (OSPF) has an assigned protocol number, making it a transport layer protocol. Similarly the Border Gateway Protocol (BGP) uses a port number from the TCP header for data flow and control.

In theory, all upper layer protocols could use either UDP or TCP. Both provide a transport layer function. The reliability requirements of the applications dictate which transport layer is used. UDP provides an unreliable, connectionless transport service, while TCP provides a reliable, in sequence connection-oriented service.

The remaining sections in the text explain the various Internet, Transport and Application layer protocols referred to in the last few pages.

TCP/IP Protocol Stack Based on Data Flow



7. Internet Protocol (IP)

IP is a connectionless protocol that is primarily responsible for addressing and routing packets between network devices. Connectionless means that a session is not established before data is exchanged.

IP is quite unreliable because packet delivery is not guaranteed. IP makes what is termed a 'best effort' attempt to deliver a packet. Along the way a packet may be lost, delivered out of sequence, duplicated or delayed.

An acknowledgement is not required when data is received. The sender or receiver is not informed when a packet is lost or out of sequence. The acknowledgement of packets is the responsibility of a higher-layer transport protocol, such as the Transmission Control Protocol (TCP).

IP is also responsible for fragmenting and reassembling packets. A large packet must be divided into smaller pieces when it has to traverse a network that supports a smaller packet size. For example, an IP packet on a Fibre Distributed Data Interface (FDDI) network may be up to 8,968 bytes long. If such a packet needs to traverse an Ethernet network, it must be split up into IP packets which are a maximum of 1500 bytes long.

Internet Protocol (IP)

- **Provides logical 32-bit network addresses**
- **Routes data packets**
- **Connectionless protocol**
 - No session is established
- **"Best effort" delivery**
- **Reliability is responsibility of higher-layer protocols and applications**
- **Fragments and reassembles packets**

Routing of IP Packets

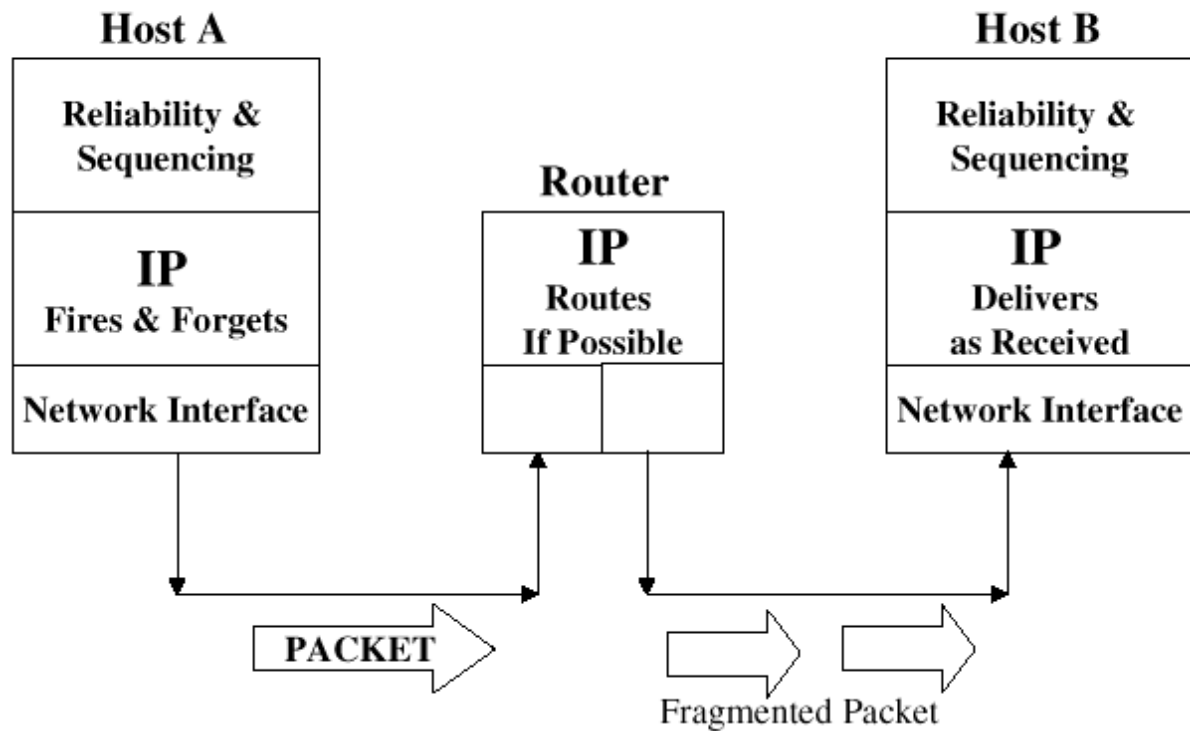
IP delivers its packets in a connectionless mode. It does not check to see if the receiving host can accept data. Furthermore it does not keep a copy in case of errors. IP is therefore said to "fire and forget".

When a packet arrives at a router, the router forwards the packet only if it knows a route to the destination. If it does not know the destination, it drops the packet. In practice routers rarely drop packets, because they typically have default routes defined. The router does not send any acknowledgements to the sending device.

A router analyses the checksum. If it is not correct then the packet is dropped. It also decreases the Time-To-Live (TTL), and if this value is zero then the packet is dropped. If necessary the router fragments larger packets into smaller ones and sets flags and Fragment Offset fields accordingly. Finally, a new

checksum is generated due to possible changes in TTL, flags and Fragment Offset. The packet is then forwarded.

The Internet Protocol

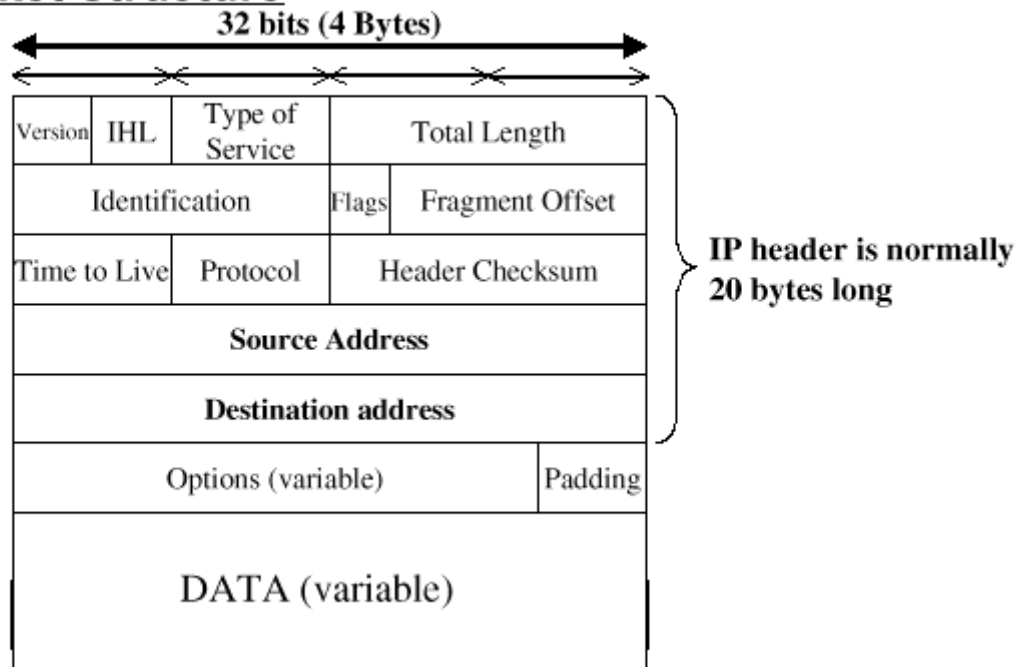


IP Packet Structure

Table 3.1 IP Packet Structure

Packet Element	Bits	Purpose
Version	4	Specifies version of the IP protocol, and hence format of the IP header being used, for example IPv4 or IPv6. This field can also be used with IPsec.
Internet Header Length (IHL)	4	Length of header in 32-bit words. Minimum value is five, which is the most common header. Header must be at least 20 bytes long.
Type of Service	8	Indication of the quality of service requested for the IP packet. It specifies reliability, precedence, delay and throughput parameters. Typically not used
Total length	16	Total packet length, including header and data, in bytes.
Ident.	16	Unique number assigned by the sending device to aid in reassembling a fragmented packet. Primary purpose is to allow the destination device to collect all fragments from a packet, since they will all have the same identification number.
Flags	3	Provides the fragmentation control fields. First bit is not used and is always 0. If second bit is 0, it means 'May fragment'. If it is 1, it means 'Don't fragment'. If the third bit is 0, it means 'Last fragment'. If it is 1, it means 'More fragments'.
Fragment Offset	13	Used with fragmented packets to aid in reassembling the full packet. The value is the number of 8-byte pieces (header bytes are not counted) that are contained in earlier fragments. In the first fragment, or in a unique fragment, this value is always zero.
Time to Live	8	Contains time(s), that packet is allowed to remain on an internetwork. Each IP device that the packet passes through will decrease the value by the time it takes it to process the IP header. All routers must decrease this value by a minimum of one. If value drops to zero the packet is discarded. This guarantees that packets cannot travel around an IP network in a loop, even if routing tables become corrupt.
Protocol	8	Indicates the higher level protocol to which IP should deliver the data in the packet, for example, UDP is 17 and TCP is 6.
Header Checksum	16	This is a checksum on the header only, which ensures integrity of header values. Sending IP device performs a calculation on the bits in the IP header, excluding the header checksum field, and places the result in the header checksum field.
Source Address	32	This is the 32-bit IP address of the sending device.
Dest. Address	32	This is the 32-bit IP address of the receiving device.
Options	Var	These are not required in every packet. They are mainly used for network testing or debugging.
Data	Var	The total length of the data field plus header is a maximum of 65535 bytes.

IP Packet Structure



Type of Service (TOS)

The Type Of Service (TOS) field indicates the way in which a packet should be handled, and is broken down into five subfields as shown in the diagram.

Three precedence bits specify packet precedence, with values ranging from 0 (normal precedence) through 7 (network control), allowing senders to indicate the importance of each packet.

Normally most host and router software ignores the TOS. However, if all hosts and routers honour precedence, TOS provides a mechanism that can allow control information to have precedence over data.

It is, for example, possible to implement congestion control algorithms that are not affected by the congestion which they are trying to control.

Bits D, T, and R specify the type of transport the packet desires.

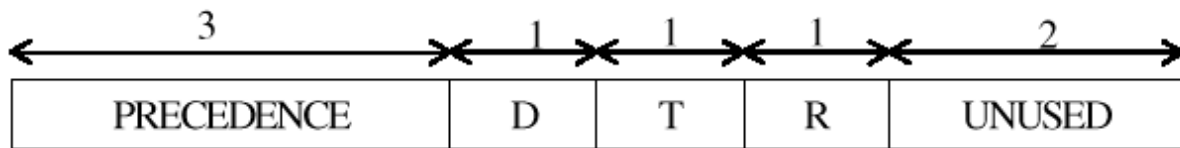
The bits request the following:

- When set, the D bit requests low delay
- The T bit requests high throughput
- The R bit requests high reliability

If a router knows more than one possible path to a given destination, it can use the Type of Transport field to select one with characteristics most similar to those desired. Suppose, for example, a router can select between a low capacity leased line or a high-bandwidth (but high-delay) satellite connection.

Packets carrying key strokes from a user to a remote computer could have the D bit set requesting that they be delivered as quickly as possible, while packets carrying a bulk file transfer could have the T bit set requesting that they travel across the high-capacity satellite path.

Type of Service (TOS)



D = Delay

T = Throughput

R = Reliability

Fragmentation

Each physical network imposes some maximum transmission size, called the

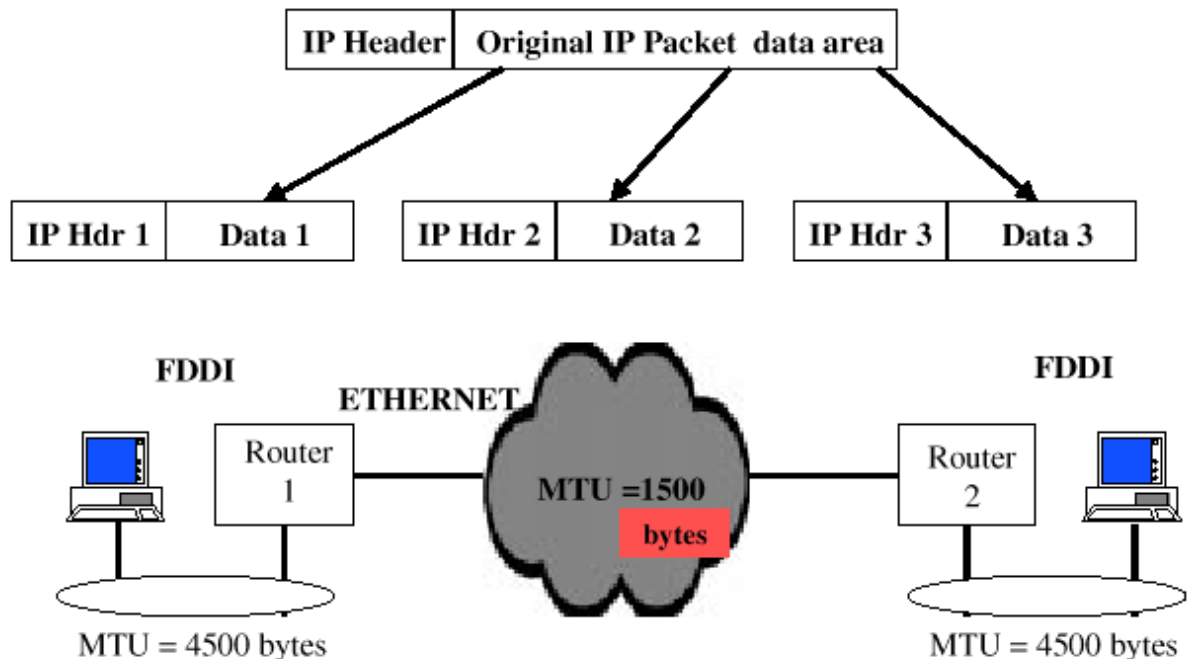
Maximum Transfer Unit (MTU), on the packets that may be sent over it. When the size of the packet exceeds the limits of the network on the outgoing interface, the packet must be broken into smaller packets, each of which carries a portion of the original data. This process is called Fragmentation.

The fragmented IP packets have data copied from the original packet into their data area. Each fragment contains an IP header that duplicates the original header, with the exception of the information in the checksum, flags and offset fields. They are treated as normal IP packets while being transported to their destination. The fragment packets may take different routes to their final destination.

When the fragment packets arrive at their destination, the destination host must join the fragments together again before processing the original packet in the normal way. If, however, one of the fragments gets lost then the complete IP packet is considered to be lost. This is because IP does not provide any acknowledgement mechanism. The remaining fragments will simply be discarded by the destination host.

Note that if a packet has a flag set to 'don't fragment' and the router decides to send this packet over a medium which does not support the size of the packet, then the packet is dropped.

Fragmentation



8. The IP Address

Every network interface on a TCP/IP device is identified by a globally unique IP address. Host devices, for example, PCs, typically have a single IP address.

Routers typically have two or more IP addresses, depending on the number of interfaces they have.

Each IP address is 32 bits long and is composed of four 8-bit fields, called octets. The address is normally represented in 'dotted decimal notation' by grouping the four octets and representing each one in decimal form. Each octet represents a decimal number in the range 0-255.

For example, 11000001 10100000 00000001 00000101, is known as 193.160.1.5.

Each IP address consists of a network ID and a host ID. The network ID identifies the systems that are located on the same network. The network ID must be unique to the internetwork. The host ID identifies a TCP/IP network device (or host) within a network. The address for each host must be unique to the network ID. In the example above, the PC is connected to network '193.160.1.0' and has a unique host ID of '.5'.

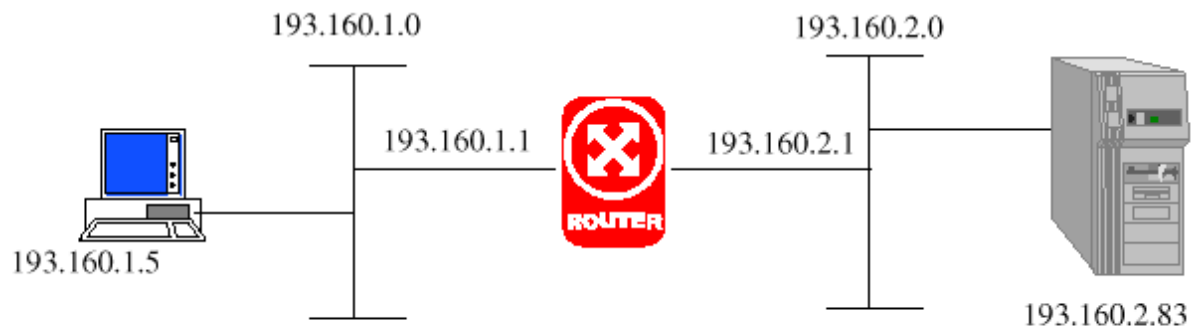
Note that all Internet addresses are assigned by a central authority. The Internet Assigned Numbers Authority (IANA) has ultimate control over network IDs assigned and sets the policy. The IANA has delegated this responsibility to three regional Internet registries:

- ARIN (American Registry for Internet Numbers)
- RIPE (Reseaux IP European)
- APNIC (Asia Pacific Network Information Centre)

Internet service providers (ISPs) apply to their regional Internet registry to get

blocks of addresses, referred to as address space. The ISPs assign addresses from those address spaces to their customers, for example, companies that want to connect to the Internet.

The IP Address



Binary Format	11000001 10100000 00000001 00000101
Dotted Decimal Notation	193.160.1.5

8.1 Traditional IP Address Classes

The first part of an Internet address identifies the network, on which a host resides, while the second part identifies the particular host on a given network.

The network-ID field can also be referred to as the network-number or the network-prefix. All hosts on a given network share the same network-prefix but must have a unique host-number.

There are five different address classes supported by IP addressing. The class of an IP address can be determined from the high-order (left-most) bits.

Class A (/8 Prefixes)

Class A addresses were assigned to networks with a very large number of hosts. The high-order bit in a class A address is always set to zero. The next seven bits (completing the first octet) represent the network ID and provide 126 possible networks. The remaining 24 bits (the last three octets) represent the host ID. Each network can have up to 16777214 hosts.

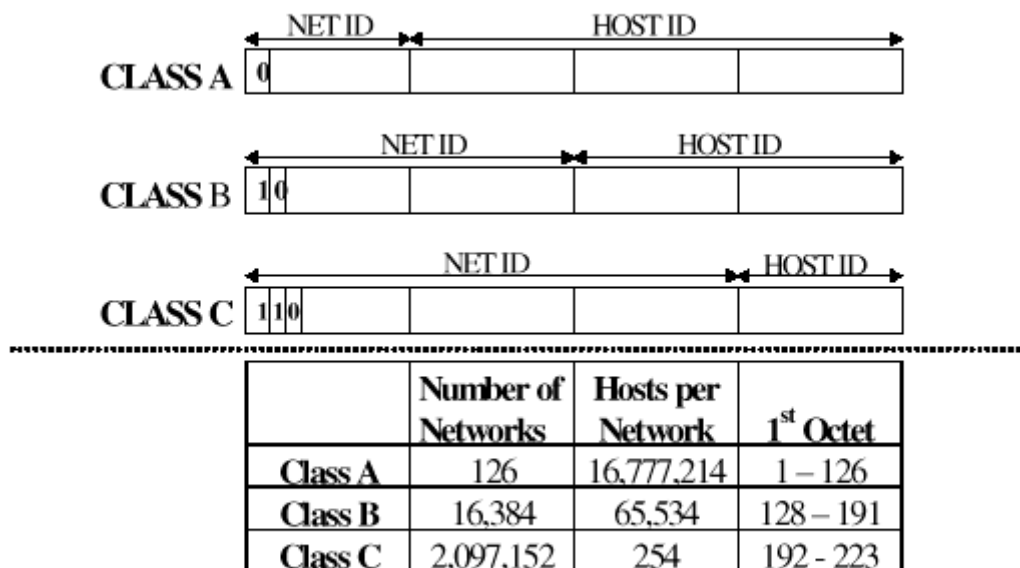
Class B (/16 Prefixes)

Class B addresses were assigned to medium-sized to large-sized networks. The two high-order bits in a class B address are always set to binary 1 0. The next 14 bits (completing the first two octets) represent the network ID. The remaining 16 bits (last two octets) represent the host ID. Therefore, there can be 16382 networks and up to 65534 hosts per network.

Class C (/24Prefixes)

Class C addresses were used for small networks. The three high-order bits in a class C address are always set to binary 1 1 0. The next 21 bits (completing the first three octets) represent the network ID. The remaining 8 bits (last octet) represent the host ID. There can, therefore, be 2097150 networks and 254 hosts per network.

Traditional IP Address Classes



2008 131/2/108 108 101/4

Class D

Class D addresses are employed for multicast group usage. A multicast group may contain one or more hosts, or none at all. The four high-order bits in a class D address are always set to binary 1 1 1 0. The remaining bits designate the specific group, in which the client participates. When expressed in dotted decimal notation, multicast addresses range from 224.0.0.0 through 239.255.255.255.

There are no network or host bits in the multicast operations. Packets are passed to a selected subset of hosts on a network. Only those hosts registered for the multicast operation accept the packet.

Some multicast group addresses are assigned as well-known addresses by the IANA. For example, the multicast address 224.0.0.6 is used for OSPF hello messages, and 224.0.0.9 is used for RIP-2.

Class E

Class E is an experimental address not available for general use. It is reserved for future use. The high-order bits in a class E address are set to 1 1 1 1 0.

Extract from RFC1812 “Requirements for IPv4 Routers”

‘The explosive growth of the Internet has forced a review of address assignment policies. The traditional uses of general purpose (Class A, B, and C) networks have been modified to achieve better use of IP’s 32-bit address space. Classless Inter Domain Routing (CIDR) is a method currently being

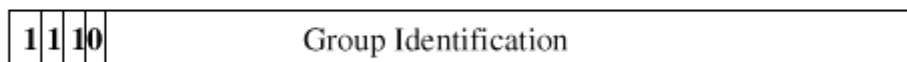
deployed in the Internet backbones to achieve this added efficiency. CIDR depends on deploying and routing to arbitrarily sized networks. In this model, hosts and routers make no assumptions about the use of addressing in the internet. The Class D (IP Multicast) and Class E (Experimental) address spaces are preserved, although this is primarily an assignment policy.'

CIDR is discussed later in this section.

Traditional IP Address Classes (continued)

● **Class D**

- **Used for multicast group usage - first 4 high-order bits are 1110**
- **1st Octet between 224 and 239**



● **Class E**

- **Reserved for future use - first 5 high-order bits are 11110**

8.2 Addressing Guidelines

The following rules must be adhered to when assigning network IDs and host IDs:

- The network ID cannot be 127. The class A network address 127.0.0.0 is reserved for loop-back and is designed for testing and inter-process communication on the local device. When any device uses the loop-back address to send data, the protocol software in the device returns the data without sending traffic across any network.
- The network ID and host ID bits of a specific device cannot be all 1s. If all bits are set to 1, the address is interpreted as a broadcast rather than a host ID. The following are the two types of broadcast:
- If a destination address contains all 1s in the network ID and the host ID (255.255.255.255) then it, is a limited broadcast, that is, a broadcast on the source's local network.
- If a destination address contains all 1s in the host ID but a proper network ID, for example, 160.30.255.255, this is a directed broadcast, that is, a broadcast on a specified network (in this example network 160.30.0.0)
- The network ID and host ID bits cannot all be 0s. If all bits are set to 0, the address is interpreted to mean 'this network only'.
- The host ID must be unique to the local network.

8.3 Private IP Address Space

RFC 1918 requests that organisations make use of the private Internet address space for hosts which require IP connectivity within the enterprise network, but do not require external connections to the global Internet. For this purpose the IANA has reserved the following three address blocks for private Internets:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

Any organisation that elects to use addresses from these reserved blocks can do so without contacting the IANA or an Internet registry. Since these addresses are never injected into the global Internet routing system, the address space can be used simultaneously by many organisations.

The disadvantage of this addressing scheme is that it requires an organisation to use a Network Address Translator (NAT) for global Internet access.

8.4 Subnet Mask

A subnet mask is a 32-bit address used to:

- Block out a portion of the IP address to distinguish the network ID from the host ID.
- Specify whether the destination host's IP address is located on a local network or on a remote network.

For example, an IP device with the configuration below knows that its network ID is 160.30.20 and its host ID is .10

Address 160.30.20.10

Subnet Mask 255.255.255.0

For convenience the subnet mask can be written in prefix length notation. The prefix-length is equal to the number of contiguous one-bits in the subnet mask. Therefore, the network address 160.30.20.10 with a subnet mask 255.255.255.0 can also be expressed as 160.30.20.10/24.

Default subnet masks or prefix lengths exist for class A, B and C addresses:

- Class A default mask 255.0.0.0 (/8)
- Class B default mask 255.255.0.0 (/16)
- Class C default mask 255.255.255.0 (/24)

Subnet Mask

- Blocks out a portion of the IP address to distinguish the Network ID from the host ID
- Specifies whether the destination's host IP address is located on a local network or on a remote network
- The source's IP address is ANDed with its subnet mask. The destination's IP address is ANDed with the same subnet mask. If the result of both ANDing operations match, the destination is local to the source, that is, it is on the same subnet.

8.5 Subnet Mask Example

ANDing is an internal process that TCP/IP uses to determine whether a packet is destined for a host on a local network, or a host on a remote network.

When TCP/IP is initialised, the host's IP address is ANDed with its subnet mask. Before a packet is sent, the destination IP address is ANDed with the same subnet mask. If both results match, IP knows that the packet belongs to a host on the local network. If the results don't match, the packet is sent to the IP address of an IP router.

To AND the IP address to a subnet mask, TCP/IP compares each bit in the IP address to the corresponding bit in the subnet mask. If both bits are 1s, the resulting bit is 1. If there is any other combination, the resulting bit is 0.

The four possible variations are as follows:

- 1 AND 1 = 1
- 1 AND 0 = 0
- 0 AND 0 = 0
- 0 AND 1 = 0

Subnet Mask Example

- For example 160.30.20.10 is on the same subnet as 160.30.20.100 if the mask is 255.255.255.0
 - Note: 1 AND 1 = 1. Other combinations = 0.

IP Address	160.30.20.10	10100000 00011110 00010100 00001010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
Result	160.30.20.0	10100000 00011110 00010100 00000000

IP Address	160.30.20.100	10100000 00011110 11001000 01100100
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
Result	160.30.20.0	10100000 00011110 00010100 00000000

8.6 Subnetting

Subnetting was initially introduced to overcome some of the problems that parts of the Internet were beginning to experience:

- Internet routing tables were becoming too large to manage.
- Local administrators had to request another network number from the Internet before a new network could be installed at their site.

Subnetting attacked the expanding routing table problem by ensuring that the subnet structure of a network is never visible outside of the organisation's private network. The route from the Internet to any subnet of a given IP address is the same, regardless of which subnet the destination host is on. This is because all subnets of a given network ID use the same network prefix, but different subnet numbers. The routers within the private organisation need to differentiate between the individual subnets, but as far as the Internet routers are concerned all of the subnets in the organisation are collected into a single routing table entry.

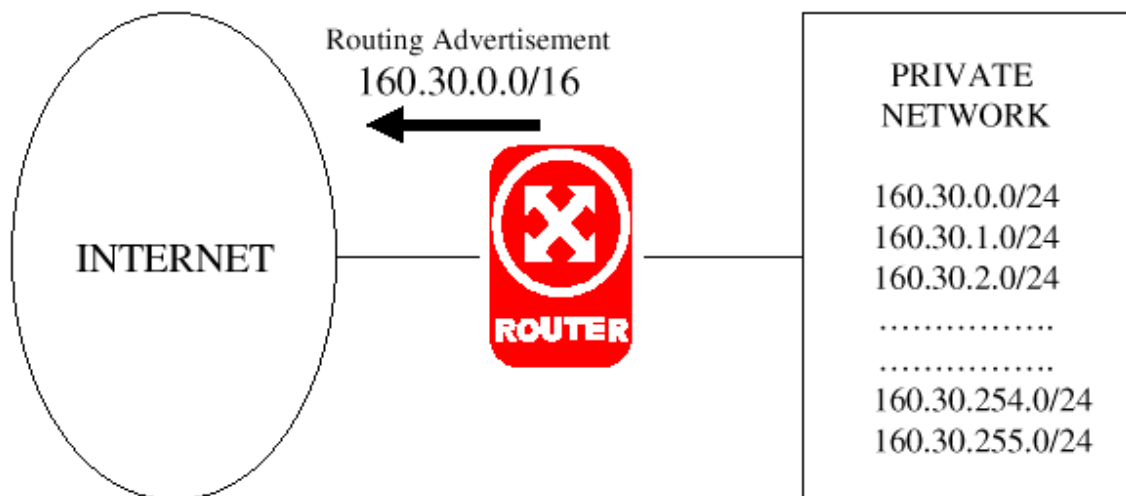
Subnetting helps to overcome the registered number issue by assigning each organisation one (or in some cases a few) network number(s) from the IPv4 address space. The organisation is then free to assign a distinct subnetwork number to each of its internal networks. This allows the organisation to deploy additional subnets without needing to obtain a new network number from the Internet.

For example, a site with several logical networks uses subnet addressing to cover them with a single 'class b' network address. The router accepts all traffic from the Internet addresses to network 160.30.0.0, and forwards traffic to the internal subnetworks based on the third octet of the classful address.

The deployment of subnetting within the private network provides several benefits:

- The size of the global Internet routing table does not grow because the site administrator does not need to obtain additional address space, and the routing advertisements for all of the subnets are combined into a single routing table entry.
- The local administrator has the flexibility to deploy additional subnets without obtaining a new network number from the Internet.
- Rapid changing of routes within the private network does not affect the Internet routing table, since Internet routers do not know about the reachability of the individual subnets. They just know about the reachability of the parent network number.

Subnetting



- Before subnetting: 1 network with approx.. 65 thousand hosts
- After subnetting: 256 networks with 254 hosts per subnet

8.7 Network with Customised Mask

The example shown in the diagram calculates the number of subnets available when a customised mask is applied.

The IP address space 160.30.0.0/16 has been allocated to an organisation. Using the default subnet mask on a 'class b' network gives one single network with a total of 65534 hosts.

Using the customised mask 255.255.255.0, the organisation has up to 256 subnets, rather than just one single network.

A shortcut method of working out the number of subnets is:
(2 to the power of the number of bits in the subnet mask, excluding the default mask portion).

In the example this is 2^8 , which gives a total of 256 subnets.

Example: Network with Customised Mask

Allocated IP address space 160.30.0.0/16
--

3-octet mask 255.255.255.0				
8 bits available for subnets and 8 bits available for host				
	255	255	255	0
	1111 1111	1111 1111	1111 1111	0000 0000
<i>No. of Subnets</i>	Network			Host
160.30.0.x	1010 0000	0001 1110	0000 0000	xxxx xxxx
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">↓</div> <div style="text-align: center;">↓</div> </div>				
160.30.255.x	1010 0000	0001 1110	1111 1111	xxxx xxxx
Maximum of 256 subnets (2^8)				

This is the same example as the other, but this time we want to calculate the number of hosts in any one of the 256 subnets.

The host addresses 0 and 255 cannot be used. Therefore the lowest possible host address on each subnet is 1, and the highest possible host address on each subnet is 254.

A shortcut method of working out the number of hosts in a subnet is:
 {(2 to the power of the number of zeros in the mask) less two}.
 In the example this is $(2^8)-2$, which gives a total of 254 hosts.

Example: Network with Customised Mask (continued)

Allocated IP address space 160.30.0.0/16

3-octet mask 255.255.255.0				
8 bits available for subnets and 8 bits available for host				
	255	255	255	0
	1111 1111	1111 1111	1111 1111	0000 0000
No. of hosts	Network			Host
160.30.x.1	1010 0000	0001 1110	xxxx xxxx	0000 0001
<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="text-align: center;">↓</div> <div style="text-align: center;">↓</div> </div>				
160.30.x.254	1010 0000	0001 1110	xxxx xxxx	1111 1110
Maximum of 254 hosts ($2^8 - 2$)				

Subnetting Example

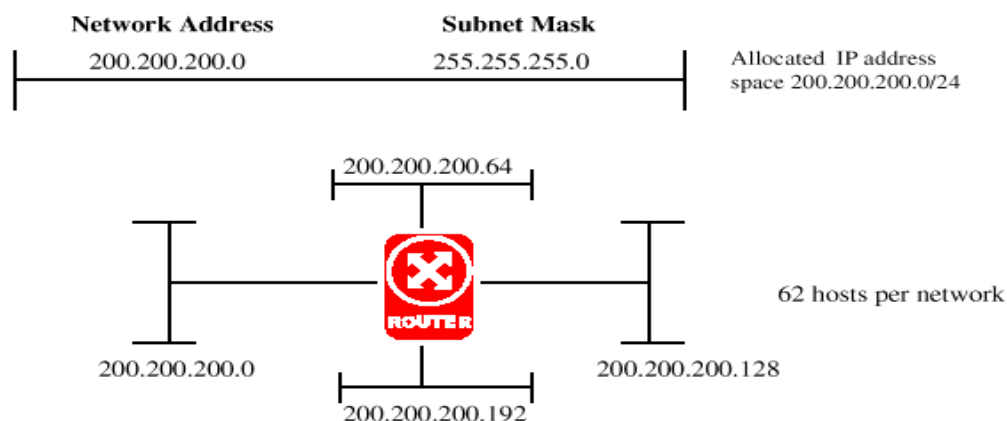
In the example shown in the diagram, a small company has been assigned the IP address space 200.200.200.0/24.

Without subnetting, up to a maximum of 254 hosts can share this network. In this configuration, if one device sends out an IP broadcast (e.g. DHCP Discover message), the broadcast is received by every device on the network.

To improve performance, the network administrator may reduce the number of devices that receive the broadcast by splitting the network into smaller subnets separated by a router.

In the example, the network has been split into four smaller subnets with a maximum of 62 hosts on each subnet.

Subnetting Example



Note: Subnet mask for each subnet = 255.255.255.192

9. Network with VLSM

Variable Length Subnet Masks (VLSM) support more efficient use of an organisation's assigned IP address space. One of the major problems with the earlier limitation of supporting only a single subnet mask across a given network-prefix was that once the mask was selected, it locked the organisation into a set number of fixed size subnets.

For example, assume that a network administrator decided to configure the 200.200.200.0/24 with a /26 extended-network-prefix (subnet mask). This permits four subnets, each of which supports a maximum of 62 devices.

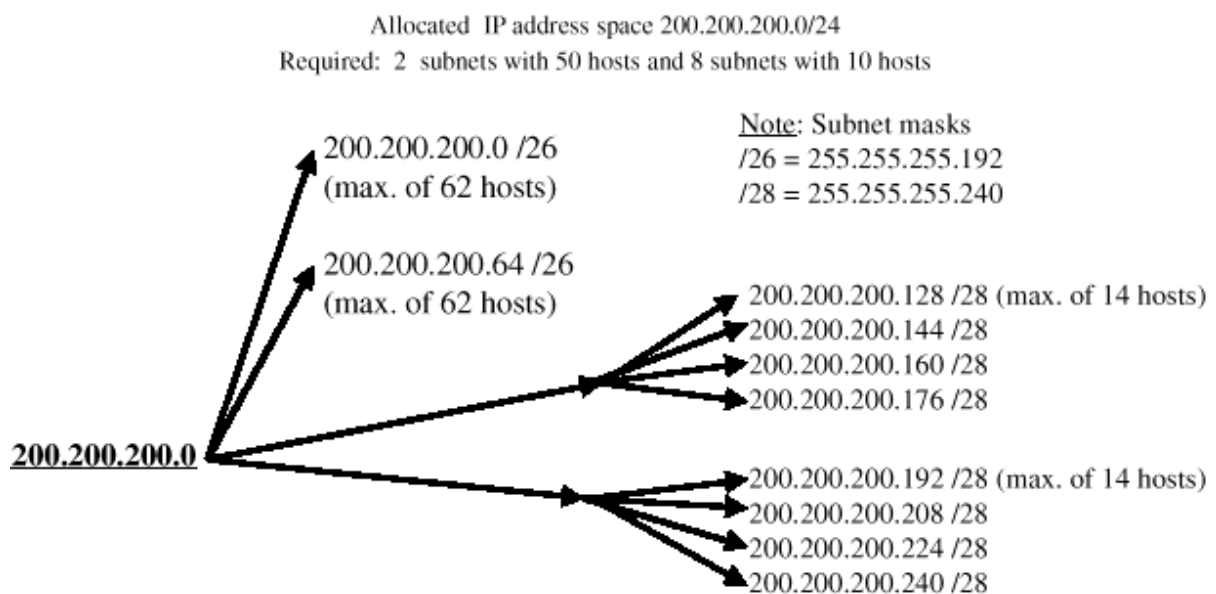
Alternatively, if we configure with a /28 extended-network-prefix then this permits 16 subnets with 14 hosts each.

Neither of these are suitable if we want 2 subnets with 50 hosts and 8 subnets with 10 hosts. If the /26 mask is used throughout the network then there are not enough subnets. If the /28 mask is used throughout the network then there are not enough hosts' addresses for two of the subnets.

The solution to this problem is VLSM, which allows a subnetted network to be assigned more than one subnet mask. In this example, VLSM allows us to use both a /26 mask and a /28 mask.

We use the /26 mask to produce two subnets with a maximum of 62 devices each. We use the /28 mask to produce eight subnets with a maximum of 14 host each. This is suitable for our stated requirements.

Example Network with VLSM



9.1 Example: Network with VLSM

The diagram shows an example of a portion of a real network with VLSM implemented.

The company owns the block of addresses 160.40.0.0/16. On Site A all devices are on the same subnet. There can be a maximum of 1022 devices ($2^{10}-2$),

since there are 10 bits available for host addresses. Valid network addresses are from 160.40.144.1 to 160.40.147.254.

Similarly, on Site C all devices are on the same subnet. There can be a maximum of 1022 devices. Valid network addresses are from 160.40.148.1 to 160.40.151.254.

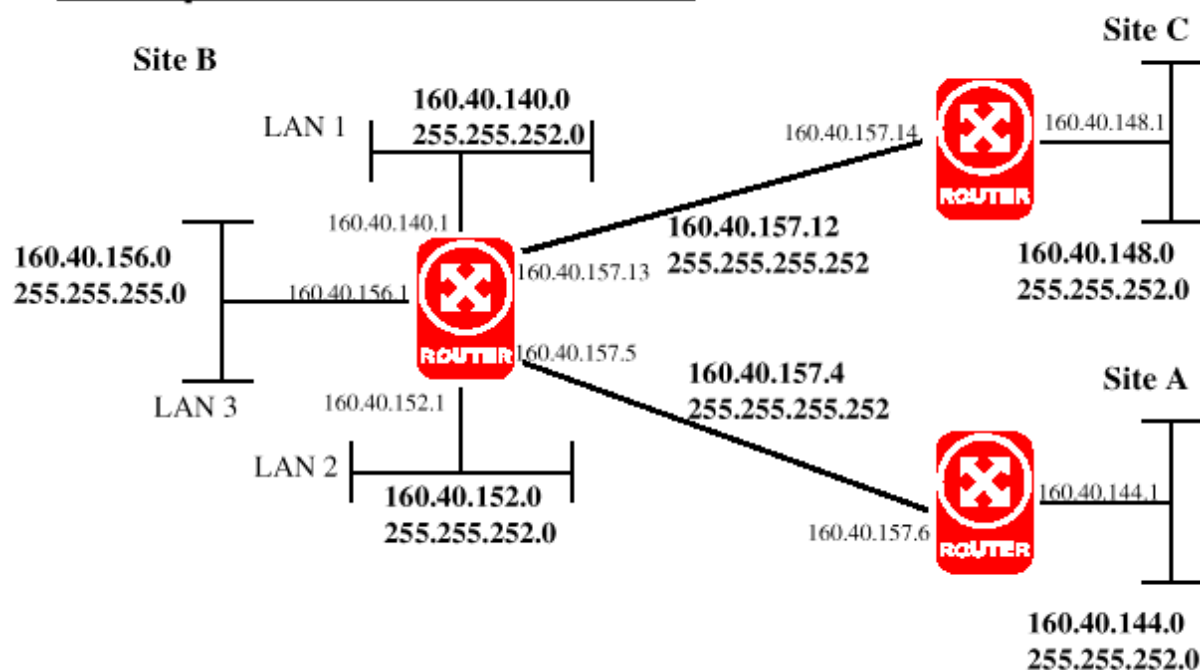
On Site B there are three subnets. Two of the subnets (LAN 1 & LAN 2) can have 1022 devices.

Valid network addresses on LAN 1 are 160.40.140.1 to 160.40.143.254.
Valid network addresses on LAN 2 are 160.40.152.1 to 160.40.155.254.

Also on Site C there is the address space 192.80.156.10/24, which can have a maximum of 254 devices. Valid network addresses are 192.80.156.1 to 192.180.156.254.

Both the WAN links use the smallest possible subnets to support 2 network addresses by using a mask 255.255.255.252.

Example Network with VLSM



9.2 Variable Length Subnets from 1 to 16

The table in the diagram lists the variable length subnets from 1 to 16, the Classless Inter Domain Routing (CIDR) representation and the dotted decimal equivalents.

The use of network addresses and subnet masks is in the past, although the language used to describe them remains in current use. They have been replaced by the more manageable network prefix, in a system known as CIDR.

A network prefix is, by definition, a contiguous set of bits at the more significant end of the address that defines a set of systems. Host numbers select among those systems.

The classical IP addressing architecture used addresses and subnet masks to

discriminate the host number from the network address. With network prefixes, it is sufficient to indicate the number of bits in the prefix. Both classical IP addressing and network prefixes are in common use. Architecturally correct subnet masks are capable of being represented using the prefix length description. Routers always treat a route as a network prefix, and reject configuration and routing information inconsistent with that model.

Referring to the table, we can see that a /15 allocation can also be specified using the traditional dotted-decimal mask notation of 255.254.0.0. Also a /15 allocation contains a bit-wise contiguous block of 131,070 IP addresses, which can be classfully interpreted as two 'class b' networks or 512 'class c' networks.

Variable Length Subnets from 1 to 16

CIDR Prefix-length	Subnet Mask	# Individual Addresses	# Classful Networks
/1	128.0.0.0	2048 M	128 A
/2	192.0.0.0	1024 M	64 A
/3	224.0.0.0	512 M	32 A
/4	240.0.0.0	256 M	16 A
/5	248.0.0.0	128 M	8 A
/6	252.0.0.0	64 M	4 A
/7	254.0.0.0	32 M	2 A
/8	255.0.0.0	16 M	1 A or 256 Bs
/9	255.128.0.0	8 M	128 B
/10	255.192.0.0	4 M	64 B
/11	255.224.0.0	2 M	32 B
/12	255.240.0.0	1 M	16 B
/13	255.248.0.0	524,286	8 B
/14	255.252.0.0	262,142	4 B
/15	255.254.0.0	131,070	2 B
/16	255.255.0.0	65,534	1 B or 256 Cs

9.3 Variable Length Subnets from 17 to 30

The table in the diagram lists the variable length subnets from 17 to 30, the CIDR representation and the dotted decimal equivalents.

In the CIDR model, each piece of routing information is advertised with a bit mask (or prefix-length). The prefix-length is a way of specifying the number of the leftmost contiguous bits in the network-portion of each routing table entry.

For example, a network with 20 bits of network-number and 12 bits of host-number is advertised with a 20-bit prefix length (a /20). The clever thing is that the IP address advertised with the /20 prefix could be a former class A, B or C.

Routers that support CIDR do not make assumptions based on the first 3-bits of the address. Instead, they rely on prefix-length information provided with the route.

In a classless environment, prefixes are viewed as a bit-wise contiguous block

of the IP address space. For example, all prefixes with a /20 prefix represent the same amount of address space (2^{12} or 4,094 host addresses). Furthermore a /20 prefix can be assigned to a traditional class A, B or C network number.

For example, each of the following /20 blocks represent 4094 host addresses: 10.23.64.0/20, 130.5.0.0/20, 200.7.128.0/20.

Note that the number of individual addresses, in the diagram, does not include the all-zeros address and the all-ones address. For example, if we use the /30 prefix (255.255.255.252 mask) then we have only two possible addresses in the subnet (and not four).

Variable Length Subnets from 17 to 30

CIDR Prefix-length	Subnet Mask	# Individual Addresses	# Classful Networks
/17	255.255.128.0	32,766	128 Cs
/18	255.255.192.0	16,382	64 Cs
/19	255.255.224.0	8,190	32 Cs
/20	255.255.240.0	4,094	16 Cs
/21	255.255.248.0	2,046	8 Cs
/22	255.255.252.0	1,022	4 Cs
/23	255.255.254.0	510	2 Cs
/24	255.255.255.0	254	1 C
/25	255.255.255.128	126	1/2 C
/26	255.255.255.192	62	1/4 C
/27	255.255.255.224	30	1/8 C
/28	255.255.255.240	14	1/16 C
/29	255.255.255.248	6	1/32 C
/30	255.255.255.252	2	1/64 C

10. CIDR Route Aggregation

CIDR supports route aggregation, where a single routing table entry can represent the address space of perhaps thousands of traditional classful routes. This allows a single routing table entry to specify how to route traffic to many individual network addresses. Route aggregation helps control the amount of routing information in the Internet's backbone routers, reduces route flapping (rapid changes in route availability) and eases the local administrative burden of updating external routing information.

In the example shown in the diagram assume that an Internet Service Provider (ISP) owns the address block 200.25.0.0/16 . This block represents 65536 (2^{16}) IP addresses (or 256 /24s). From the 200.25.0.0/16 block the ISP wants to allocate the 200.25.16.0/20 address block. This smaller block represents 4,096 (2^{12}) IP addresses (or 16 /24s).

In a classful environment the ISP is forced to cut up the /20 address block into

16 equal size pieces. However, in a classless environment the ISP is free to cut up the address space any way it wants. It could slice up the address space into 2 equal pieces and assign one portion to company A, then cut the other half into 2 pieces (each 1/4 of the address space) and assign one piece to company B, and finally slice the remaining fourth into 2 pieces (each 1/8 of the address space) and assign one piece each to company C and company D. Each of the individual companies is free to allocate the address space within its 'Intranetwork' as it sees fit. A prerequisite for aggregating networks' addresses is that they must be consecutive and fall on the correct boundaries. For example, we cannot aggregate 200.25.24.0/24, 200.25.26.0/24, 200.25.27.0/24 without including the address space 200.25.25.0/24.

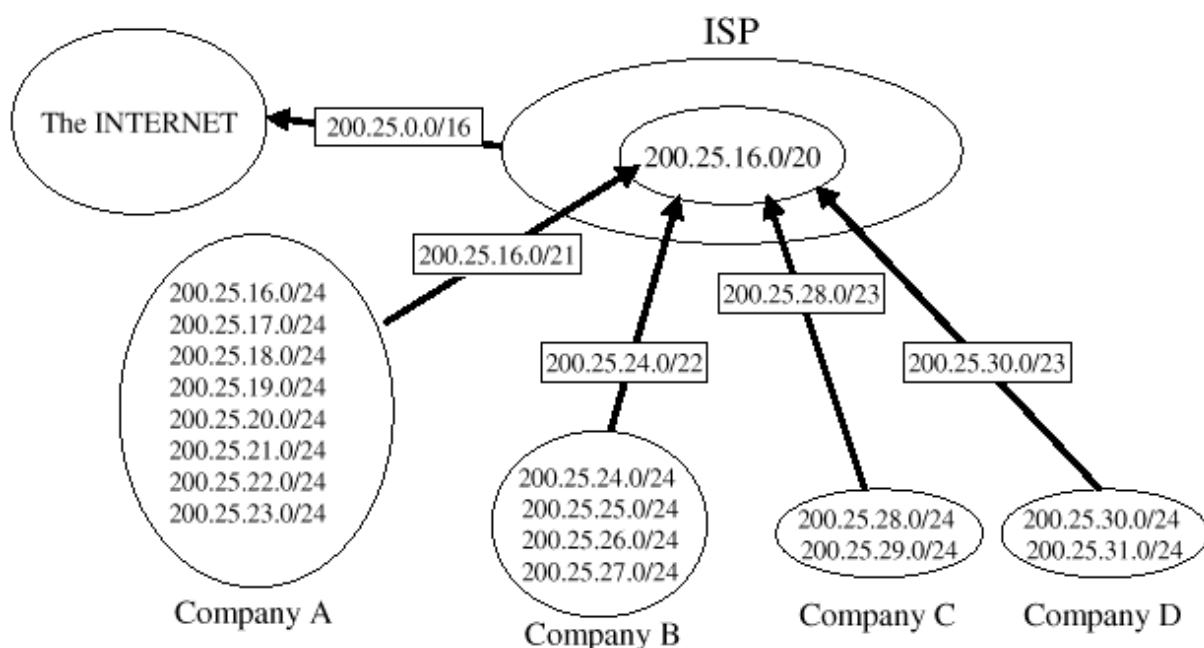
CIDR plays an important role in controlling the growth of the Internet's routing tables. The reduction of routing information requires that the Internet be divided into addressing domains. Within a domain, detailed information is available about all the networks that reside in the domain. Outside an addressing domain, only the common network prefix is advertised. This allows a single routing table entry to specify a route to many individual network addresses.

The diagram illustrates how the allocation described above helps reduce the size of the Internet routing tables.

- Company A aggregates 8 /24s into single advertisement (200.25.16.0/21).
- Company B aggregates 4 /24s into single advertisement (200.25.24.0/22).
- Company C aggregates 2 /24s into single advertisement (200.25.28.0/23).
- Company D aggregates 2 /24s into single advertisement (200.25.30.0/23).

Finally the ISP is able to inject the 256 /24s in its allocation into the Internet with a single advertisement - 200.25.0.0/16.

CIDR Route Aggregation



10.1 Subnet ID Tables

The subnet ID table shows all the most common subnet IDs.

Take, as an example, an allocation of the address block 160.30.0.0/16 by the IANA. Assume that we require large subnets with approximately 1500 devices per subnet. We first consult the variable length subnet table to decide on the subnet mask. The mask of 255.255.248.0 is suitable as it gives subnets each containing 2046 devices. Then by consulting the subnet ID table we can see that the different subnet IDs for this mask are:

160.30.**0.0**, 160.30.**8.0**, 160.30.**16.0**, 160.30.**24.0**,..... 160.30.**240.0**, 160.30.**248.0**.

Alternatively, assume that we wanted small subnets with approximately 50 devices per subnet. This time, from the subnet conversion table, we can see that the mask 255.255.255.192 is suitable because it gives subnets with 62 devices each (64 addresses including the all-zeros and all-ones addresses).

Then by consulting the subnet ID table we can see that the different subnet IDs for this mask are:

160.30.0.**0**, 160.30.0.**64**, 160.30.0.**128**, 160.30.0.**192**, 160.30.1.**0**, 160.30.1.**64**, 160.30.1.**128**, 160.30.1.**192**,160.30.255.**0**, 160.30.255.**64**, 160.30.255.**128**, 160.30.255.**192**.

Subnet ID Tables

No. of Bits in Mask	Subnet Mask	Subnet IDs	
16	255.255.0.0	0	3rd Octet
17	255.255.128.0	0, 128	
18	255.255.192.0	0, 64, 128, 192	
19	255.255.224.0	0,32,64,96,128,160,192,224	
20	255.255.240.0	0,16,32,48,64,80,96,112,128,144,160,176,192,208,224,240	
21	255.255.248.0	0,8,16,24,32,40,48,56,64,.....,216,224,232,240,248	
22	255.255.252.0	0,4,8,12,16,20,24,28,32,.....,236,240,244,248,252	
23	255.255.254.0	0,2,4,6,8,10,12,14,16,18,.....,246,248,250,252,254	
24	255.255.255.0	0,1,2,3,4,5,6,7,8,9,10,11,.....,251,252,253,254,255	4th Octet
25	255.255.255.128	0, 128	
26	255.255.255.192	0, 64, 128, 192	
27	255.255.255.224	0,32,64,96,128,160,192,224	
28	255.255.255.240	0,16,32,48,64,80,96,112,128,144,160,176,192,208,224,240	
29	255.255.255.248	0,8,16,24,32,40,48,56,64,.....,216,224,232,240,248	
30	255.255.255.252	0,4,8,12,16,20,24,28,32,.....,236,240,244,248,252	

11. IP Addresses and Symbolic Names

Each computer is assigned an Internet Protocol address, which appears in each IP packet sent to that computer. Anyone who has used the Internet knows that users do not need to remember or enter IP addresses.

Computers are also assigned symbolic names. Application software allows a user to enter one of the symbolic names when identifying a specific computer.

Although symbolic names are convenient for humans, they are inconvenient for computers. The underlying network protocols only understand addresses, so some mechanism to map symbolic names to IP addresses is required.

11.1 Domain Name Resolution

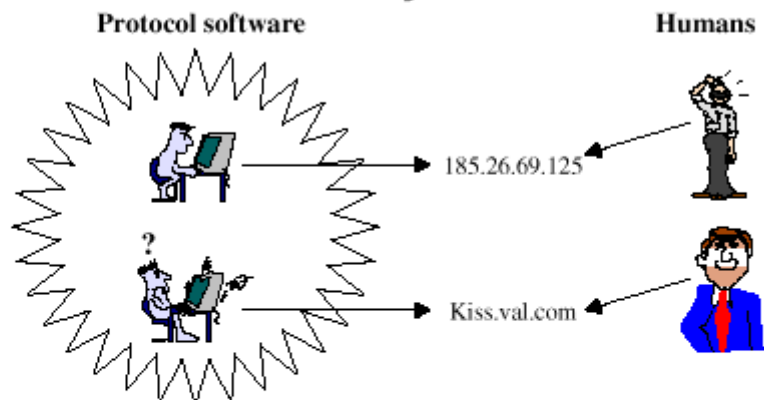
Application software translates symbolic computer names into equivalent Internet addresses. A database for implementing the naming scheme is distributed across the internet. This method of mapping the symbolic names to

IP addresses through a distributed database is known as the Domain Name System (DNS).

There are two ways to use the domain name system: by contacting name servers one at a time or asking the name server system to perform the complete translation. In either case, the client software forms a domain name query that contains the name to be resolved and a code that specifies whether the name server should translate the name completely or supply the name of another server that can translate the name. It sends the query to a name server for resolution.

When a domain server receives a query, it checks to see if the name lies in the subdomain for which it is an authority. If so, it translates the name to an address according to its database, and appends an answer to the query before sending it back to the client. If the name server cannot resolve the name completely, it checks to see what type of interaction the client specified. If the client requested complete translation (recursive resolution), the server contacts a domain name server that can resolve the name and returns the answer to the client. If the client requested non-recursive resolution (iterative resolution), the name server cannot supply an answer. It generates a reply that specifies the name server the client should contact next to resolve the name.

DNS - Domain Name System



- **Internet addresses are hard for humans to remember**

- Easy for protocol software to work with.

- **Symbolic names are more natural for humans**

- Hard for protocol software to work with.

11.2 Domain Name System

The Domain Name System (DNS) is based on a hierarchical scheme, with the most significant part of the name located on the right. The segment on the left is the name of the individual computer. Other segments in a domain name identify the group that owns the name. For example, the Burger Department at Pizza has the domain name:

Burger.Krusty.cookie.Pizza.ie

Basically, the Internet is divided into hundreds of top-level domains where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned and so forth. There are two types of top-level domains, generic and countries.

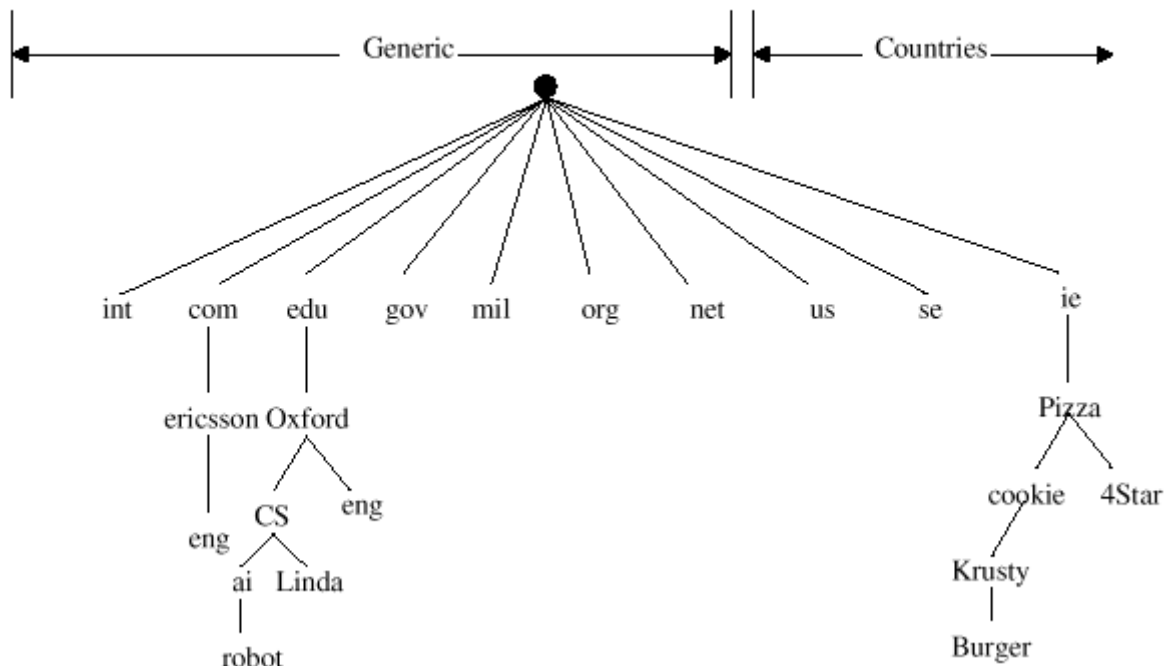
The generic domains are represented by a three letter entry:

- **com** (Commercial organisation)
- **edu** (Educational institution)
- **gov** (Government organisation)
- **mil** (Military group)
- **net** (Major network support centre)
- **org** (Organisation other than those above)
- **int** (International organisation)

The country domains include a two letter entry for every country, as defined in ISO 3166. For example, .ie = Ireland.

Each domain is named by the path upward to the top-level domain. The segments are separated by periods. For example, the Ericsson engineering department may have the domain name: eng.ericsson.com.

Internet Domain Name Space



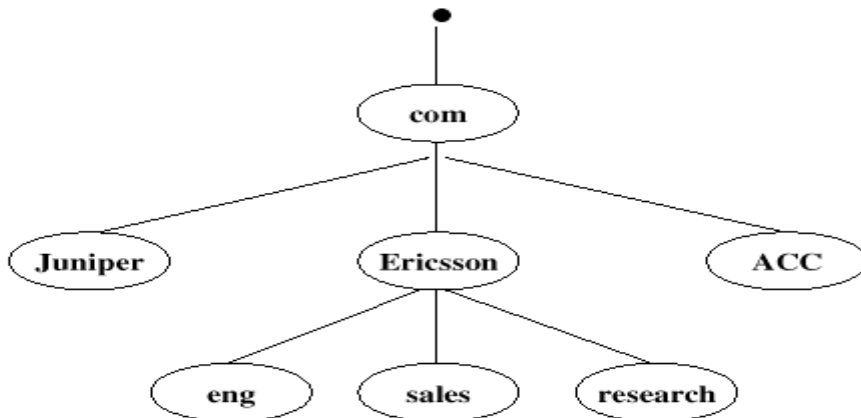
11.3 Resolving a Name

The translation of a domain name into an equivalent IP address is called name resolution. The name is said to be resolved to an address. A host asking for DNS name resolution is called a resolver.

Each resolver is configured with the address of a local domain name server. If a resolver wishes to become a client of the DNS server, the resolver places the specified name in a DNS request message and then sends the message to the local server. The resolver then waits for the server to send a DNS reply message that contains the answer. When communicating with a DNS server resolvers are configured to use the User Datagram Protocol (UDP), because it requires less overhead for a single request. When an incoming request to the server specifies a name for which a server is an authority, the server answers the request directly. That is, the server looks up the name in its local database and then sends a reply to the resolver.

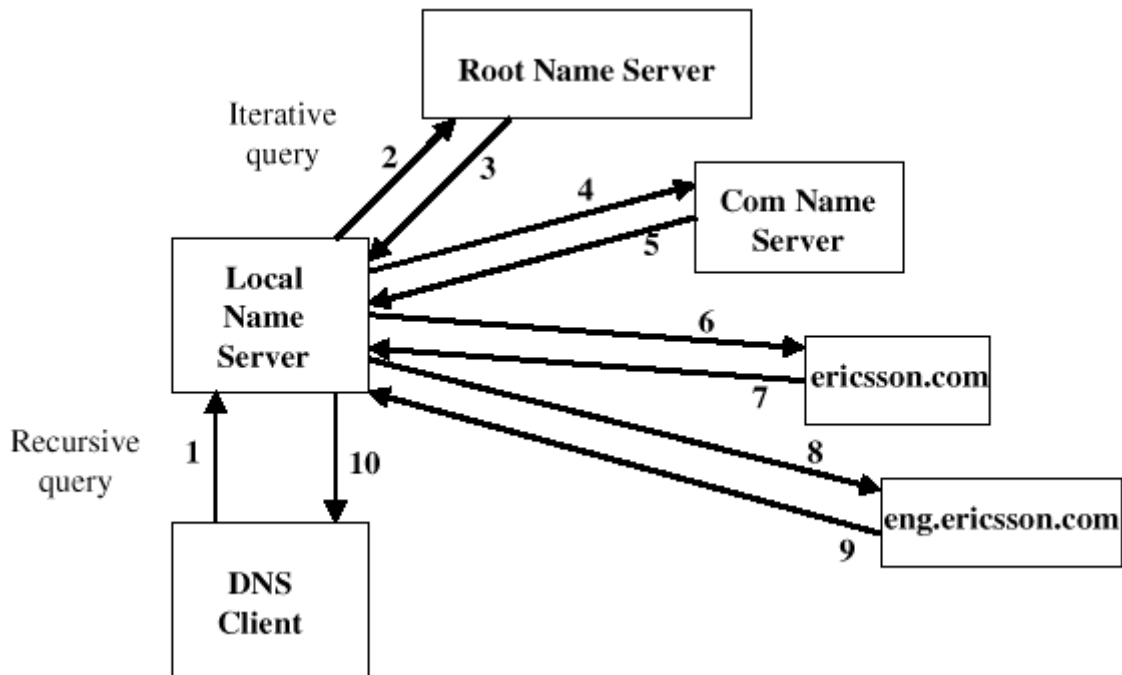
If, however, a request arrives for a name outside the set for which the server is an authority, further client-server interaction results. The server temporarily becomes a client of another name server. When the second server returns an answer, the original server sends a copy of the answer back to the resolver from which the request arrived.

Domain Name Resolution



How does a DNS server know which other DNS server is the authority for a given name? The answer is that it does not know. Each server, however, knows the address of a root server. Knowing the location of a root server is sufficient because the name can be resolved from there.

1. The resolver (DNS client) sends a recursive DNS query to its local DNS server asking for the IP address of 'server1.eng.ericsson.com'. The local name server is responsible for resolving the name and cannot refer the resolver to another name server.
2. The local name server is not an authority for the name so it sends an iterative query, for server1.eng. ericsson.com, to a root name server.
3. The root name server has authority for the root domain and replies with the IP address of a name server for the com top-level domain.
4. The local name server sends an iterative query for 'server1.eng.ericsson.com' to the com name server.
5. The com name server replies with an IP address for the name server servicing the ericsson.com domain.
6. The local name server sends an iterative query for 'server1.eng.ericsson.com' to the ericsson.com name server.
7. The ericsson.com name server replies with an IP address for the name server servicing the eng. ericsson.com domain.
8. The local name server sends an iterative query for 'server1.eng.ericsson.com' to the eng.ericsson.com name server.
9. The eng. ericsson.com name server replies with the IP address corresponding to server1.eng. ericsson.com (or an indication that no such name exists).
10. The local name server sends the IP address of 'server1.eng.ericsson.com' to the original resolver.



11.4 DNS Caching

Internet name servers use name caching to reduce the traffic on the internet and to improve performance. Each server maintains a cache of recently used names as well as a record of where the mapping information for that name was obtained.

When a client asks the server to resolve a name, the server first checks to see if it has authority for that name. If not, the server checks its cache to see if the name has been resolved recently. Servers report cached information to clients, but mark it as a non-authoritative binding. The binding is the domain name and corresponding IP address. The local DNS server sends the client the domain name of the server (S) from which they obtained the binding. The local server also sends along additional information that tells the client the binding between S and an IP address, so that the client knows how to contact S if required.

Therefore, clients receive answers quickly but the information may be out-of-date. If efficiency is important then the client will choose to accept the non-authoritative answer and proceed. If accuracy is important the client will choose to contact the authority and verify that the binding between name and address is still valid.

To keep the cache correct, servers time each entry and dispose of entries that exceed a reasonable time. Servers allow the authority for an entry to configure this time-out. When an authority responds to a request, it includes a Time To Live (TTL) value in the response that specifies how long the authority guarantees the binding to remain. Authorities can thus reduce network overhead by specifying long time-outs for entries that they expect to remain unchanged, while improving correctness by specifying short time-outs for entries that they expect to change frequently.

DNS Caching

- ◆ **Internet name servers use name caching to reduce the traffic on the Internet and improve performance**
- ◆ **Servers report cached information to clients, but mark it as a non-authoritative binding**
- ◆ **If efficiency is important, the client will choose to accept the non-authoritative answer and proceed**
- ◆ **If accuracy is important the client will choose to contact the authority and verify that the binding between name and address is still valid**
- ◆ **Whenever an authority responds to a request, it includes a Time To Live (TTL) value in the response that specifies how long it guarantees the binding to remain**

12. Address Resolution Protocol (ARP)

Network devices must know each other's hardware address in order to communicate on a network. Address resolution is the process of mapping a host's IP address to its hardware address.

The Address Resolution Protocol (ARP) is responsible for obtaining hardware addresses of TCP/IP devices on broadcast-based networks.

ARP uses a local broadcast of the destination IP address to acquire the hardware address of the destination device. Once the hardware address is obtained, both the IP address and the hardware address are stored as one entry in the ARP cache for a period of time. This is called a dynamic entry. The ARP cache is always checked for an IP address/hardware address mapping before initiating an ARP request broadcast.

An alternative to dynamic entries is to use static entries. In this case the IP address/hardware address mapping is manually entered into the ARP cache.

Static entries reduce broadcast traffic on the network. The disadvantage of static entries is that they are time consuming to implement and if either the IP or the hardware address of a remote device changes, the entry in the ARP cache will be incorrect and thus prevent the two devices from communicating.

Address Resolution Protocol (ARP)

- **A source must know a destination's hardware address before it can send an IP packet directly to it**
- **ARP is the mechanism that maps IP to hardware addresses**
- **ARP uses a local broadcast to obtain a hardware address dynamically**
- **ARP stores mappings in cache for future use**
- **Static entries can be manually entered into the ARP cache**

12.1 ARP Request

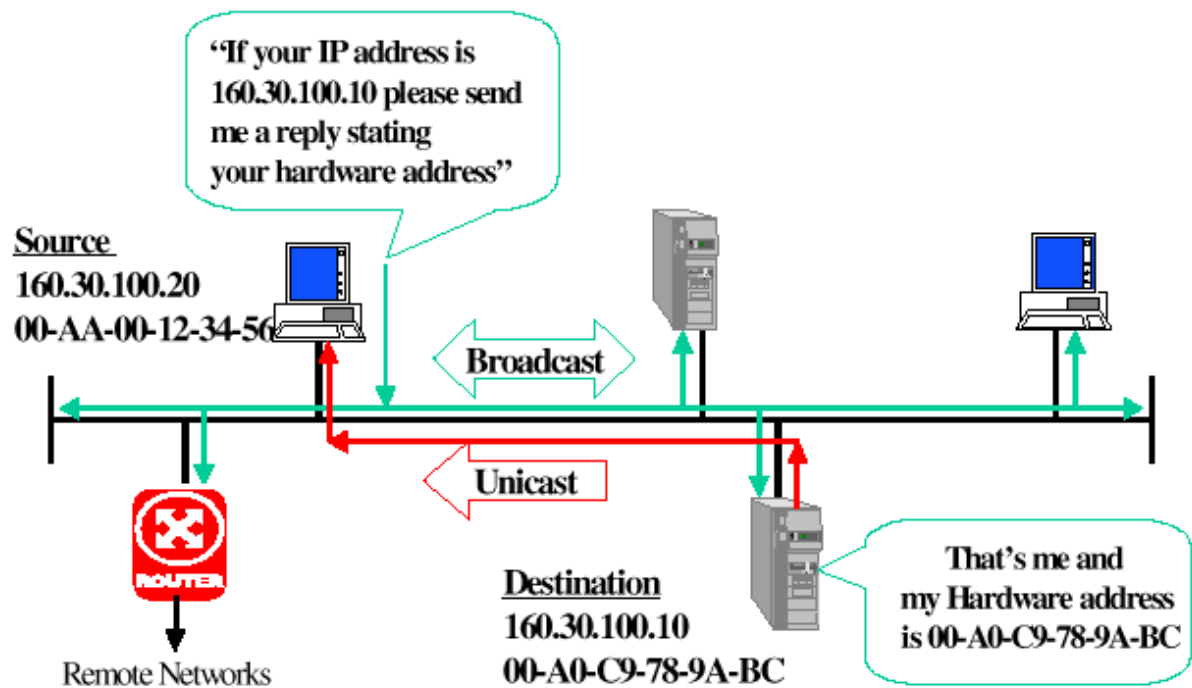
The source device knows its own IP and hardware address and the IP address of the device that it wants to send the information to. It checks its existing ARP cache for the hardware address of the destination host. If no mapping is found, the source builds an ARP request packet, looking for the hardware address to match the IP address.

The ARP request is a broadcast so all local devices receive and process it.

Each device checks for a match with its own IP address. The destination device determines that there is a match and sends an ARP reply directly to the source device with its hardware address. Both devices update their ARP cache with the IP address/hardware address mapping of the other device. From then on the devices can communicate directly with each other. If devices do not communicate with each other after a period of time they will clear the entry from their ARP caches.

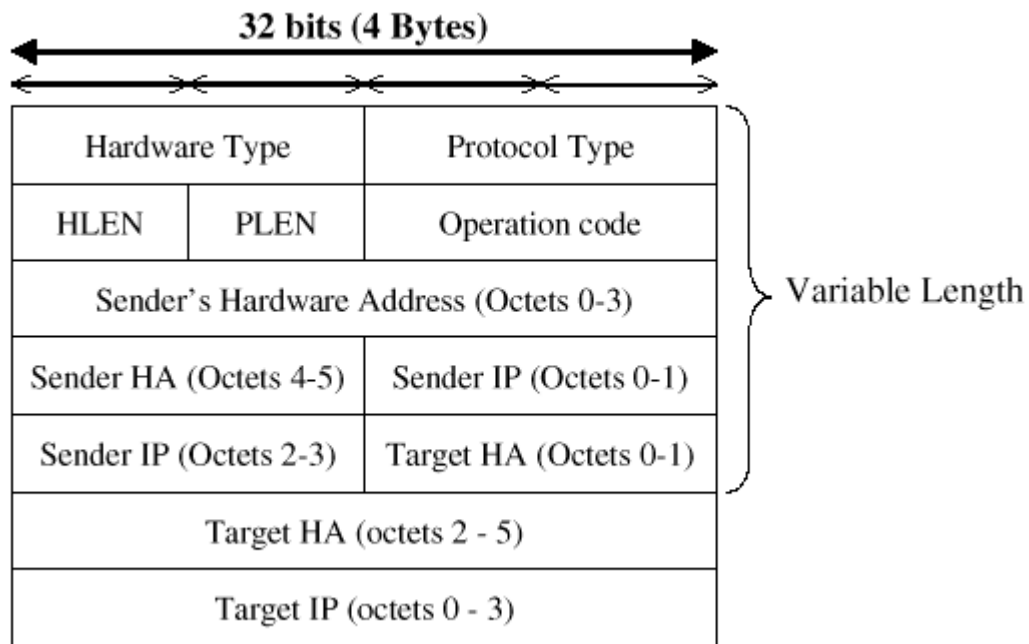
Note that if the destination host is on a remote network, the IP software determines the IP address of a locally attached next-hop router to which to send the IP packet. The sending device then uses ARP to obtain the hardware address of the local router (not the remote destination host)

Address Resolution Protocol (ARP)



ARP Packet Structure:

Packet Element	Number of Bits	Purpose
Hardware Type	16	This specifies the hardware interface type, for example, Ethernet has a value of 1.
Protocol Type	16	This specifies the higher-level protocol whose address needs to be mapped onto the hardware, for example, IP - 0800.
HLEN, hardware address length	8	This specifies the length in bytes of the hardware address in this packet, for example, Ethernet - 6.
PLEN, protocol address length	8	This specifies the length in bytes of the protocol address in this packet. For IP this is four.
Operation code	16	This specifies whether this is an ARP request (1) or an ARP reply (2).
Sender's hardware address	48	This contains the hardware address of the sender (the ARP requester).
Sender's IP address	32	This contains the protocol address of the sender (the ARP requester).
Target's hardware address	48	This contains the hardware address of the target (the ARP responder).
Target's IP address	32	This contains the protocol address of the sender (the ARP responder).



13. Reverse ARP

ARP solves the problem of mapping a host's IP address to its hardware address, but sometimes the reverse problem must be solved. Reverse ARP (RARP) is used when the hardware address is given, for example an Ethernet address, but not its corresponding IP address.

The RARP protocol allows a newly booted device to broadcast its Ethernet address and say: 'My 48-bit Ethernet address is 00-A0-C9-78-9A-BC. Does anyone know my IP address?'. The RARP protocol uses the same message format as ARP. The server sees this request, looks up the Ethernet address in its configuration files and sends back the corresponding IP address. This type of server is known as an RARP server.

To prevent multiple servers from sending a reply simultaneously, thus causing collisions, a primary server may be designated for each host wishing to use RARP. This server replies immediately, and all non-primary servers simply listen and note the time of the request. If the primary server is unavailable, the originating node will time out and re-broadcast the RARP request. The non-primary servers respond when they hear a copy of the request within a short time after the original broadcast.

For example, printers use RARP to obtain an IP address.

Note that RARP requests stay within the local LAN, so the servers must also reside there.

Reverse Address Resolution Protocol

- **Reverse ARP is the mechanism that maps hardware addresses to the IP address**
- **RARP protocol allows a newly booted machine to broadcast its Ethernet address**
- **The RARP server sees this request and sends back the corresponding IP address**

14. Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) reports errors and sends control messages on behalf of IP.

ICMP does not attempt to make IP a reliable protocol. It simply attempts to report errors and provide feedback on specific conditions. ICMP messages are carried as IP packets and are therefore unreliable.

Message Element	Bits	Purpose
Type	8	Specifies the type of ICMP message.
Code	8	Specifies the condition inside one type of ICMP message.
Checksum	16	A checksum carried out on the ICMP header only.
Identifier	16	Used by the sender to match replies to requests.
Sequence number	16	Used by the sender to match replies to requests.
Optional Data	-	Contains information to be returned to the sender.

- Reports errors and sends control messages on behalf of IP
- ICMP messages are encapsulated within an IP packet
- One of the most frequently used debugging tools uses ICMP
- **ICMP Message Format**

<i>IP Header.....</i>		
Type	Code	Checksum
Identifier		Sequence Number
Optional Data		

14.1 ICMP Message Types

Although each has its own format, all ICMP messages begin with the same three fields:

- An 8-bit integer message type field identifying the message
- An 8-bit code field providing further information about the message type
- A 16-bit checksum field

Destination Unreachable

When a router cannot forward or deliver an IP packet, it sends a destination unreachable ICMP message back to the original source.

Source Quench Message

A host or router uses source quench messages to report congestion to the original source and to request it to reduce its current rate of packet transmission.

Redirect Message

When a router detects a host using a non-optimal route, it sends the host an ICMP redirect message, requesting that the host change its routes. The router also forwards the original packet onto its destination.

Time Exceeded Message

Whenever a router discards a packet because its hop count has reached zero or because a timeout occurred while waiting for fragments of a packet, it sends an ICMP time exceeded message back to the packet's source.

Parameter Problem Message

When a router or host finds problems with a packet not covered by previous ICMP error messages (for example, an incorrect packet header), it sends a parameter problem message to the original source.

Timestamp Request and Timestamp Reply Messages

One method of clock synchronisation in TCP/IP networks is to send a timestamp request message to another device. The receiving device returns a timestamp reply back to the device making the request.

Address Mask Request and Address Mask Reply Messages

To learn the subnet mask used for the local network, a device can send an address mask request message to a router (via unicast or broadcast) and receive an address mask reply.

ICMP Message Types

TYPE FIELD	ICMP Message Types
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
8	Echo Request
11	Time exceeded for a packet
12	Parameter problem on a packet
13	Timestamp request
14	Timestamp reply
15	Information request (obsolete)
16	Information reply (obsolete)
17	Address mask request
18	Address mask reply

14.2 Echo Request and Reply Messages

One of the most frequently used debugging tools invokes the ICMP echo request and echo reply messages. A host or router sends an ICMP echo request message to a specified destination. Any device that receives an echo request formulates an echo reply and returns it to the original sender. If the sender does not receive a reply it means that the destination is unreachable.

The request also contains an optional data area. The reply contains a copy of the data sent in the request.

On many devices, the command that users invoke to send an ICMP echo request is named ping (packet internet groper). Sophisticated versions of ping send a series of ICMP echo requests, capture responses, and provide statistics about packet loss

Echo Request and Reply Message Format

<i>IP Header.....</i>		
Type = 8 (or 0)	Code = 0	Checksum
Identifier		Sequence Number
Optional Data		

These messages test whether a destination is reachable and responding, by sending ICMP echo requests and receiving back ICMP echo replies.

This test is carried out by using the “PING” command.

Unreachable Messages

When a router cannot forward or deliver an IP packet, it sends a destination unreachable message back to the original source. The code field in the destination unreachable message contains an integer that further describes the problem, as shown in the diagram.

Network unreachable errors usually imply routing failures, while host unreachable errors imply delivery failures. Destinations may be unreachable for the following reasons:

- Hardware is temporarily out of service.
- The sender specified a non-existent destination address.
- The router does not have a route to the destination network.

Most of the messages are self explanatory. For example, if the packet contains a source route option (list of routers which the packet must pass through) with an incorrect route then it may trigger a source route failed message. If a router needs to fragment a packet but the ‘don’t fragment’ (DF) bit is set, then the router sends a fragmentation needed message back to the source.

Reports of Unreachable Destinations

Code Value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and DF set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Communication with destination network administratively prohibited
10	Communication with destination host administratively prohibited
11	Network unreachable for type of service
12	Host unreachable for type of service

14.3 Traceroute

Traceroute is an application that uses ICMP and the TTL field in the IP header in order to make visible the route that IP packets follow from one host to another.

When a router gets an IP packet whose TTL is either 0 or 1 the router must not forward the packet. Instead the router throws away the packet and sends an ICMP 'time exceeded' message back to the originating host. The key to traceroute is that the IP packet containing this ICMP message has the router's IP address as the source address.

Traceroute operates as follows.

It sends an IP packet with a TTL of 1 to the destination host. The first router to handle the packet decrements the TTL, discards the packet, and sends back an ICMP time exceeded. This identifies the first router in the path. Traceroute then sends a packet with a TTL of 2, and the IP address of the second router is found. This continues until the packet reaches the destination host. Even though the arriving IP packet has a TTL of 1, the destination host will not throw it away and generate the ICMP time exceeded, since the packet has reached its final destination. Instead traceroute chooses a destination UDP port number with an unlikely value (larger than 30,000), making it improbable that an application at the destination is using that port. This causes the destination host's UDP module to generate an ICMP 'port unreachable' message when the packet arrives. All traceroute needs to do is to differentiate between the received ICMP messages (time exceeded versus port unreachable) to know that the packet has reached its final destination.

Traceroute

- ◆ Traceroute uses ICMP and the TTL field in the IP header, to let us see the route that IP packets follow from one host to another.
- ◆ Source sends packet with TTL set to 1
- ◆ First router sends back “time exceeded” message
- ◆ Source increments TTL counter by 1
- ◆ Second router on path sends back “time exceeded” message
- ◆ Process continues until ultimate destination send back “port unreachable” message.
- ◆ Source uses the responses to display the route to the destination

15. Purpose of TCP

TCP is a reliable, connection-oriented delivery service. Connection-oriented means that a session must be established before devices can exchange data.

Processes or applications communicate with each other by having both the sending and receiving device create end points. These end points are called sockets. An application creates a socket by specifying three items:

- The IP address of the device
- The transport protocol (TCP or UDP)
- The port the application is using

Each socket has a socket number (address) consisting of the IP address of the device and a 16-bit number, called a port. A port is used by transport protocols in order to identify to which application protocol or process they must deliver incoming messages. A port can use any number between 0 and 65535. All ‘well-known’ port numbers are below 1,024, for example:

- FTP is port 21
- Telnet is port 23
- DNS is port 53

TCP views the data stream as a sequence of octets or bytes that is divided into segments for transmission. Each segment travels across the network in a single IP packet. Reliability is achieved by assigning a sequence number to each segment transmitted. Data sent by TCP must be acknowledged by the receiver.

End-to-end flow control is implemented as follows: if the sending device is transmitting data faster than the receiving device is processing it, the receiver will send back an acknowledgement indicating that it has zero buffer space. This prevents the sender from sending any new data until the receiver is ready.

Transmission Control Protocol (TCP)

- **Connection-oriented**
- **Provides logical connections between a pair of processes:**
 - These are uniquely identified using sockets
 - Socket = IP address & port number, e.g. FTP is port 21
- **End-to-End reliable delivery**
- **Implements Flow Control**

15.1 Three-way Handshake

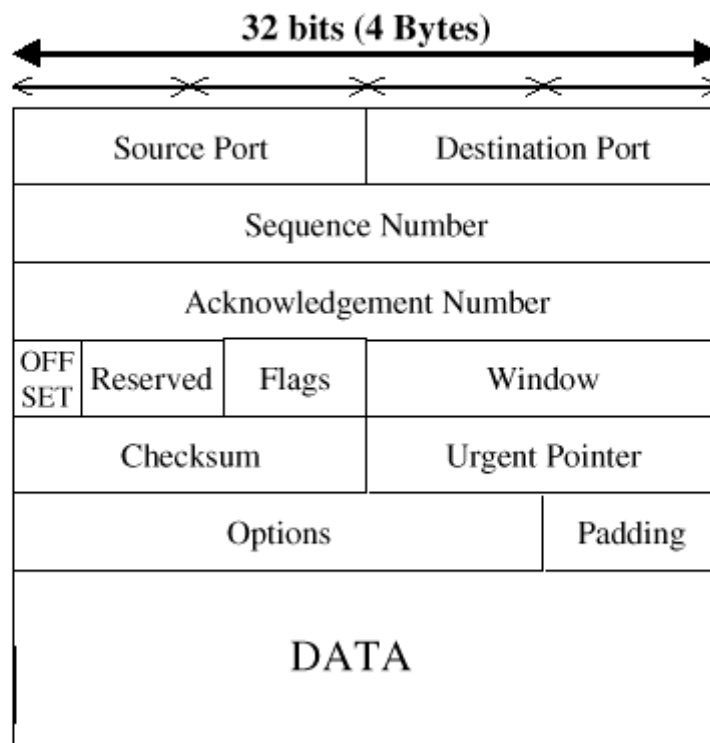
A TCP session is initialised through a three-way handshake. During this process the two communicating devices synchronise the sending and receiving of segments, inform each other of the amount of data they are able to send and receive at once (window size or buffer size) and establish a virtual connection. TCP uses a similar handshake process in order to end a connection.

Transmission Control Protocol (TCP)

- **Units of data transferred between two devices running TCP software are called “segments”**
- **Segments are exchanged to do the following:**
 - Establish a connection
 - Agree window size
 - Transfer data
 - Send acknowledgements
 - Close connection

Packet Element	Bits	Purpose
Source port	16	The TCP port number of the sending device.
Destination port	16	The TCP port number of the receiving device.
Sequence number	32	The sequence number of the data byte stream in the segment.
Acknowledgement number	32	The sequence number that the receiver expects to receive next.
Offset	4	The number of 32-bit words in the TCP header. It is needed because the Options field length is variable
Reserved	6	Reserved for future use. It must be zero.
Flags	8	These are six flags that control the behaviour of a TCP packet. They are: 1. U rgent 2. A cknowledgement 3. P ush 4. R eset connection 5. S ynchronous 6. F inish.
Window	16	Used in acknowledgement segments to implement flow control. Specifies the number of data bytes which the receiver is willing to accept.
Checksum	16	Used to verify the integrity of the TCP header. The checksum is performed on a pseudo header consisting of information obtained from the IP as well as the TCP header
Urgent Pointer	18	When urgent data is being sent (as specified in the code bits), this points to the end of the urgent data in the segment
Options	-	This is used to specify maximum segment size during the establishment of a connection.

TCP Packet Structure



15.2 Port Numbers

Every TCP segment contains the source and destination port number to identify the sending and receiving application.

TCP combines static and dynamic port binding, using a set of well known port assignments for commonly invoked programs (for example, electronic mail), while leaving most port numbers available for the operating system to allocate as programs need them. Although the standard originally reserved port numbers less than 1,024 for use as well known ports, numbers over 1,024 have now been assigned. The diagram displays some of the currently assigned TCP ports.

Well-known Port Numbers

Port Number	Description
7	Echo
20	File Transfer Protocol (FTP) data
21	File Transfer Protocol (FTP) control
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
53	Domain name server (DNS)
79	Finger
80	World Wide Web (WWW)
104	X400 Mail Sending
139	NetBIOS session service
160 -223	Reserved

15.3 Establishing a TCP Connection

In order to establish a connection, TCP uses a three-way handshake.

The client's TCP software generates a sequence number (1000 in this example). The client then requests a session by sending out a segment with the synchronisation (SYN) flag set to on. The segment header also includes the sequence number, the size of its receive buffer (window size) and the size of the biggest data segment that it can handle.

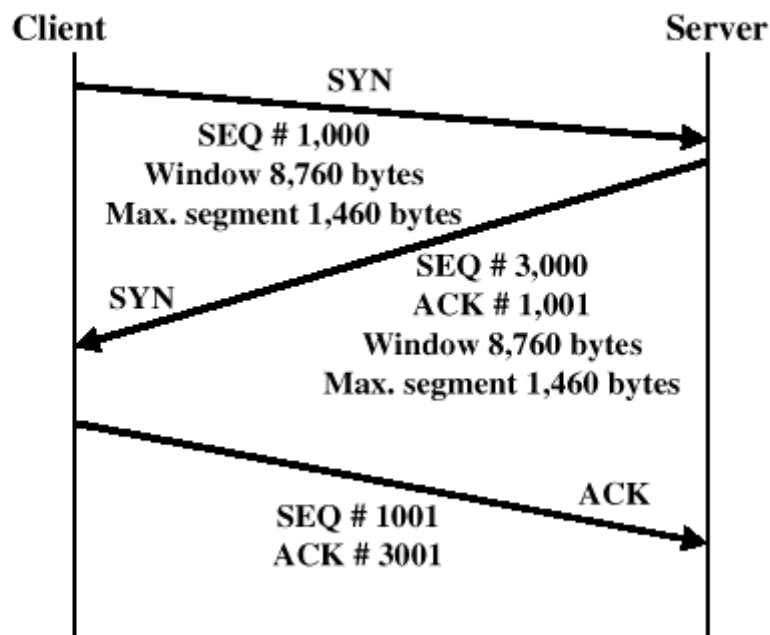
The server acknowledges (ACK) the request by sending back a segment with the synchronisation (SYN) flag set to on. The segment header contains the server's own start-up sequence number and acknowledgement as well as the number of the next segment it expects to receive from the client. The segment header also includes the size of the server's receive buffer (window size) and the size of the biggest data segment it can handle.

The client sends back an acknowledgement of the server's start-up sequence segment. It does this by sending the sequence number of the next segment it expects to receive.

Segments which carry TCP data have the push flag set to on. Some applications, for example Telnet, use the same segment to acknowledge data and transmit data. In this case both the push flag and the acknowledgement (ACK) flag are set to on.

TCP uses a similar handshake to end a connection as it does when opening a connection. In this case the finish (FIN) flag is set to on.

Establishing a TCP Connection



15.4 Positive Acknowledge-ment with Retransmit

Computers do not all operate at the same speed. Data overruns can occur when a computer sends data across a network faster than the destination is able to absorb it. Overruns can also occur in a router's buffers. Consequently data can be lost.

Several techniques are available to provide reliable delivery, and these techniques are known as flow control mechanisms.

A simple form of flow control is positive acknowledgement with retransmission. (Note: TCP does not use this mechanism. TCP uses a more complex form of acknowledgment and retransmission known as Sliding Window, which is discussed after this page.)

The positive acknowledgement with retransmission technique requires a recipient to communicate with the source, and send back an acknowledgement message when it receives data. The sender keeps a copy of each packet it sends, and waits for an acknowledgement before sending the next packet. The sender also starts a timer when it sends a packet, and retransmits the packet if the timer expires before an acknowledgement arrives. The acknowledgement will contain the sequence number that the receiver expects to receive next.

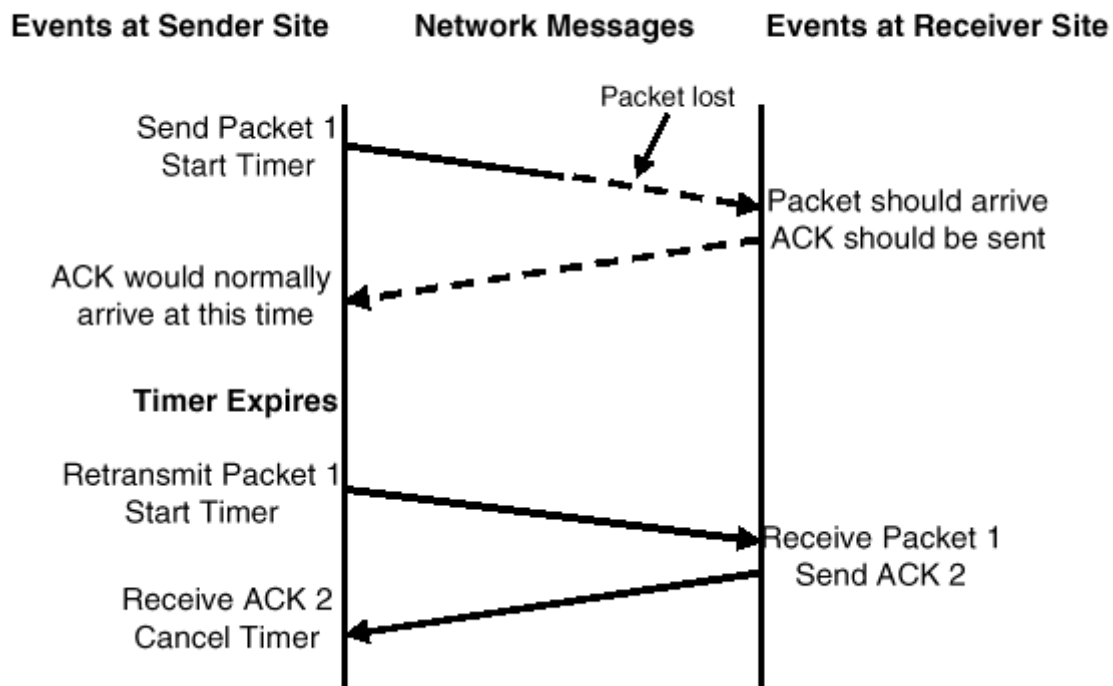
The diagram shows the events which happen when a packet is lost or corrupted. When the timer which the sender has started expires, the sender assumes that the packet was lost and retransmits it.

Problems can arise when duplicate packets are received. Duplicates can arise when networks experience long delays that cause premature retransmission.

Both packets and acknowledgements can be duplicated. In order to avoid the problem of duplication, positive acknowledgement protocols send sequence numbers back in acknowledgements. The receiver can then correctly associate

acknowledgements with packets.

Positive Acknowledgement with Retransmit



15.5 Sliding Window Protocol

In positive acknowledgement with retransmission, the sender transmits a packet and waits for an acknowledgement before transmitting another.

Data thus flows in one direction at any one time.

The network is completely idle during times that machines delay responses. As a result of this, the positive acknowledgement protocol wastes a substantial amount of network bandwidth because it must delay sending a new packet until it receives an acknowledgement for the previous packet.

The Sliding Window Protocol (SWP) uses network bandwidth more efficiently. It allows the sender to transmit multiple packets before waiting for an acknowledgement (ACK). The protocol places a small window on the sequence and transmits all packets that lie inside the window. Technically the number of packets that can be unacknowledged at any given time is constrained by the window size and is limited to a small, fixed number.

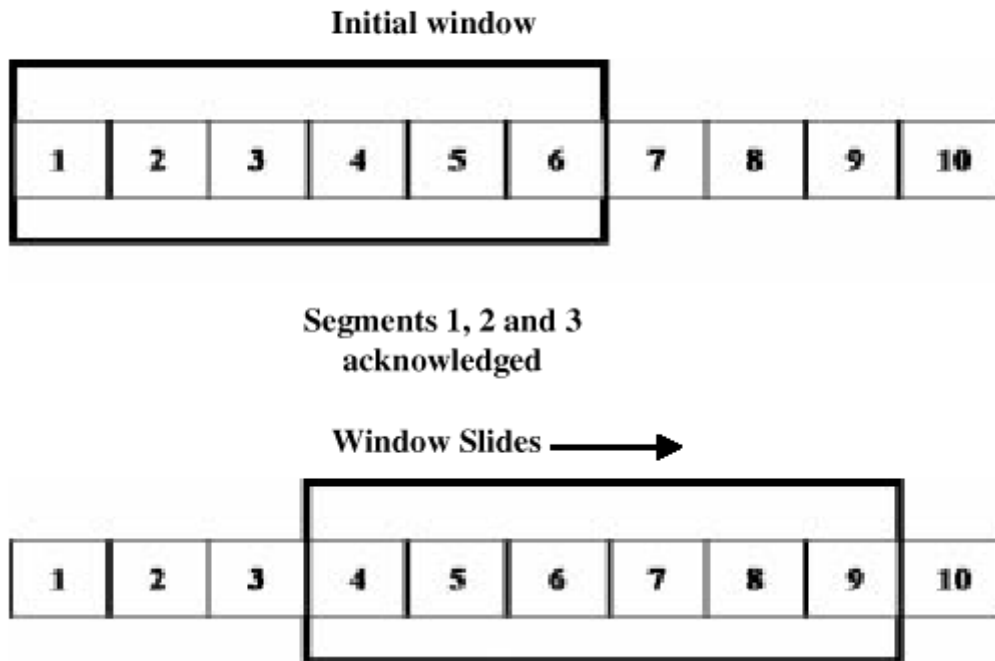
For example, in an SWP protocol with window size 6 the sender is permitted to transmit 6 packets before it receives an ACK. As the diagram illustrates, once the sender receives an acknowledgement for the first three packets inside the window, it slides the window along and sends the next packet. The window continues to slide as long as ACKs are received.

Note that the TCP sliding window mechanism operates at byte level. For example, on an Ethernet network the window size might be defined as 11680.

This means that 11680 bytes can be transmitted by the sender before it receives any acknowledgement. On an Ethernet network this is the equivalent

of eight TCP segments filled to their maximum size, assuming the TCP and IP headers are twenty bytes each.

Sliding Window Protocol



15.6 Operation of the SWP

The performance of the Sliding Window Protocol depends on the window size and the speed at which the network accepts packets. The receiver can choose how much to acknowledge, thus throttling the sender to match its capacity.

The diagram displays an example of the operation of the Sliding Window Protocol when sending three segments.

A Sliding Window Protocol keeps a separate timer for each unacknowledged segment. If a segment is lost, the timer expires and the sender retransmits that segment. When the sender slides its window, it moves past all acknowledged segments. At the receiving end, the protocol software keeps an analogous window accepting and acknowledging segments as they arrive.

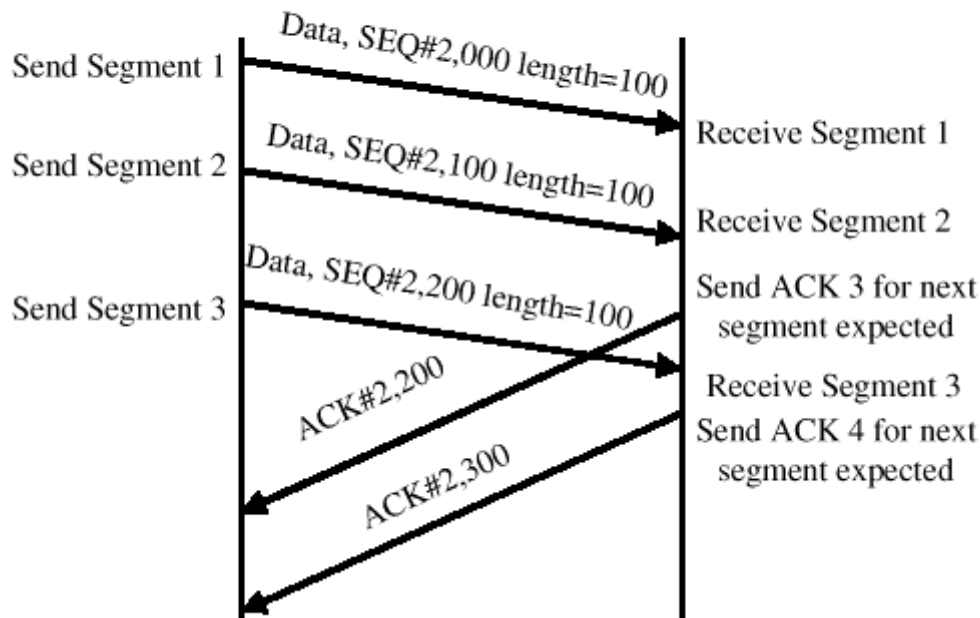
TCP allows the window size to vary over time. The advantage of this is that it provides flow control as well as reliable transfer. If the receiver's buffers begin to become full then it cannot tolerate more packets, and it sends a smaller window advertisement. In the extreme case, the receiver advertises a window size of zero to stop all transmissions. Later, when buffer space becomes available, the receiver advertises a nonzero window size to trigger the flow of data again.

Note that in TCP the acknowledgement number sent is the sequence number of the next data byte (not segment or packet) that the receiver is expecting. It is the sum of the last sequence number which it received and the length of the data, in bytes.

For example, if a device receives a segment with sequence number 2000 and a

length of 1000 bytes, it will send back an acknowledgement number of 3000.

Sliding Window Protocol



16. Slow Start Algorithm

Older versions of TCP would start a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. While 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 router must queue the packets, and it's possible for that router to run out of space. This naive approach can reduce the throughput of a TCP connection drastically.

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 acknowledgements 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 initialised to one segment (i.e. the segment size announced by the other end, or the default, typically 536 or 512). Each time an ACK is received, the congestion window is increased by one segment.

The sender can transmit up to the minimum of the congestion window and 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 may delay its ACKs, typically sending one ACK for every two segments that it receives.

At some point the capacity of the Internet can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has become too large.

Early implementations performed Slow Start only if the other end was on a different network. Current implementations always perform Slow Start.

Slow Start Algorithm

- 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 initialised to one segment
- Each time an ACK is received, the congestion window is increased by one segment
- The sender can transmit up to the minimum of the congestion window and the advertised window.
- Slow start provides an exponential growth (send one segment, then two, then four, and so on)
- The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver

16.1 Congestion Avoidance

Congestion can occur when data arrives on a big pipe (a fast LAN) and gets sent out a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. Congestion Avoidance is a way of dealing with lost packets.

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 occurring and the receipt of duplicate ACKs.

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 then 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*, and a Slow Start threshold size, *ssthresh*. The combined algorithm operates as follows:

1. Initialisation for a given connection sets *cwnd* to one segment and *ssthresh* to 65535 bytes.
2. The TCP output routine never sends more than the minimum of *cwnd* and the receiver's advertised window.
3. When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size (the minimum of *cwnd* and the receiver's advertised window, but at least two segments) is saved in *ssthresh*. Additionally, if the congestion is indicated by a timeout, *cwnd* is set to one segment (i.e., Slow Start).
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 requires that *cwnd* begin at one segment, and be 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 $\text{segsz} * \text{segsz} / \text{cwnd}$ each time an ACK is received, where *segsz* 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 RTT), whereas Slow Start increments *cwnd* by the number of ACKs received in a round-trip time.

16.2 Fast Retransmit

Modifications to the Congestion Avoidance algorithm were proposed in 1990.

The first thing to note is that TCP may generate an immediate acknowledgement (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.

Since 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.

Fast Retransmit

- TCP may generate an immediate acknowledgement (a duplicate ACK) when an out-of-order segment is received
- 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
- Since 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
- 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

16.3 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 that more than just a packet has been lost. Since 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 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.
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 should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is Congestion Avoidance, since TCP is reduced to one-half the rate it was at when the packet was lost.

17. Purpose of UDP

User Datagram Protocol (UDP) provides a connectionless packet service that offers unreliable 'best effort' delivery. This means that the arrival of packets is not guaranteed, nor is the correct sequencing of delivered packets.

UDP is used by applications that do not require an acknowledgement of receipt of data, for example, audio or video broadcasting. UDP is also used by applications that typically transmit small amounts of data at one time, for example, the Simple Network Management Protocol (SNMP).

UDP provides a mechanism that application programs use to send data to other application programs. UDP provides protocol port numbers used to distinguish between multiple programs executing on a single device. That is, in addition to the data sent, each UDP message contains both a destination port number and a source port number. This makes it possible for the UDP software at the destination to deliver the message to the correct application program, and for the application program to send a reply.

User Datagram Protocol

- Connectionless
 - No session is established
- Does not guarantee delivery
 - No sequence numbers
 - No acknowledgements
- Reliability is the responsibility of the application
- Uses port numbers as end points to communicate

The UDP header is divided into the following four 16-bit fields.

Source port

This is the UDP port number of the process on the sending device.

Destination port

This is the UDP port number of the process on the destination device.

Length

This is the size in bytes of the UDP packet, including the header and data. The minimum length is 8 bytes, the length of the header alone.

UDP Checksum

This is used to verify the integrity of the UDP header. The checksum is performed on a “pseudo header” consisting of information obtained from the IP header (source and destination address) as well as the UDP header.

The purpose of using a pseudo-header is to verify that the UDP packet has reached its correct destination. The correct destination consists of a specific machine and a specific protocol port number within that machine. The UDP header itself specifies only the protocol port number. Thus, to verify the destination, UDP on the sending machine computes a checksum that covers the destination IP address as well as the UDP packet. At the ultimate destination, UDP software verifies the checksum using the destination IP address obtained from the header of the IP packet that carried the UDP message. If the checksum agrees, then it must be true that the packet has reached the intended destination host as well as the correct protocol port within that host.

User Datagram Protocol

● UDP Packet Format

Source Port	Destination Port
Length	UDP Checksum
DATA	

● Checksum performed on Pseudo-Header

18. Purpose of BOOTP and DHCP

The Bootstrap Protocol (BOOTP) and the Dynamic Host Configuration Protocol (DHCP) are upper layer programs which automate the initial loading and configuration of hosts.

Instead of using the RARP server to obtain an IP address, a newly booted device may use these programs in order to obtain an IP address, a bootable file address, and configuration information. These programs may be used on networks that dynamically assign hardware addresses (which preclude using RARP). They also provide a centralised management of IP addresses and configuration files, which eliminates the need for per-host information files.

BOOTP and DHCP are actually the same program with different options.

BOOTP enhanced with the new options is called DHCP.

The BOOTP sequence commences with a client, for example a diskless workstation, being booted. The client initiates a BOOTP request with a broadcast address to all stations on the local network. The BOOTP request

contains the client's hardware (MAC) address. The BOOTP server receives the BOOTP requests, on UDP port 67. The server looks up the assigned IP address and puts it in the response message. It also adds the name of the BOOTP server and the name of the appropriate load file that may be executed.

Depending on the implementation, it may also add other configuration parameters such as the subnet mask and default gateway. When the client diskless workstation receives the reply (on UDP port 68) it uses the information supplied by the server to initiate a TFTP get message to the server specified.

The response to the TFTP get message is an executable load file.

BOOTP (BOOTstrap Protocol)

- A newly booted device may use BOOTP to obtain an IP address, a bootable file address, and configuration information.
 - The client initiates a BOOTP request with a broadcast address to all stations on the local network
 - The BOOTP server monitors for BOOTP requests (on UDP port 67).
 - The server looks up the assigned IP address and puts it in the response message.
 - It also adds the name of the BOOTP server and the name of the appropriate load file that may be executed.
 - It may also add other configuration parameters such as the subnet mask and default gateway.
 - The client receives the reply (on UDP port 68).
 - it uses the information supplied by the server to initiate a TFTP get message to the server specified.
 - The response to the TFTP get message is an executable load file.

● DHCP is an enhanced version of BOOTP

To keep an implementation as simple as possible, BOOTP messages have fixed length fields, and replies have the same format as request.

Field OP specifies whether the message is a request (1) or a reply (2). As in ARP, fields HTYPE and HLEN specify the network hardware type and length of the hardware address (for example, Ethernet has type 1 and address length 6).

The client places 0 in the HOPS field. If it receives the request and decides to pass the request to another device (for example, to allow boot strapping across multiple routers), the BOOTP server increments the HOPS count. The TRANSACTION ID field contains an integer that diskless machines use to match responses with requests. The SECONDS field reports the number of seconds since the client started to boot.

The CLIENT IP ADDRESS field and all the fields following it contain the most important information. To allow the greatest flexibility, clients fill in as much information as they know and leave remaining fields set to zero. For example, if a client knows the name or address of a particular server from which it wants information, it can fill in the server IP address or server host name fields. If

these fields are nonzero only the server with matching name/address will answer the request; if they are zero, any server that receives the request will reply.

BOOTP can be used from a client that already knows its IP address (for example, to obtain boot file information). A client that knows its IP address places it in the CLIENT IP ADDRESS field; other clients use zero. If the client's IP address is zero in the request, a server returns the client's IP address in the YOUR IP ADDRESS field.

The ROUTER IP ADDRESS field is set to zero by the client (0.0.0.0), and if the request is handled by a router, it records its address in the field.

The CLIENT HARDWARE ADDRESS field is used by the client for its MAC address.

The SERVER HOST NAME field is optional and may be set to zero by the client and server.

The BOOT FILE NAME may be set to zero by the client or optionally set to a generic filename to be booted. For example, 'unix'. The server will replace the field with a fully qualified path and file name of the appropriate boot file.

The VENDOR-SPECIFIC AREA CONTAINS optional information to be passed from the server to the client. This includes IP addresses of routers, time servers, and DNS servers.

BOOTP Message Format

0	8	16	24	31
OP	HTYPE	HLEN	HOPS	
TRANSACTION ID				
SECONDS		UNUSED		
CLIENT IP ADDRESS				
YOUR IP ADDRESS				
SERVER IP ADDRESS				
ROUTER IP ADDRESS				
CLIENT HARDWARE ADDRESS (16 OCTETS)				
SERVER HOST NAME (64 OCTETS)				
BOOT FILE NAME (128 OCTETS)				
VENDOR-SPECIFIC AREA (64 OCTETS)				

DHCP centralises and manages the allocation of TCP/IP configuration information by automatically assigning IP addresses to devices configured to use DHCP. Implementing DHCP eliminates some of the configuration problems associated with manually configuring TCP/IP. Typing in the IP address, subnet mask, or default gateway incorrectly can lead to problems including communication difficulties and network problems due to a duplicate IP address.

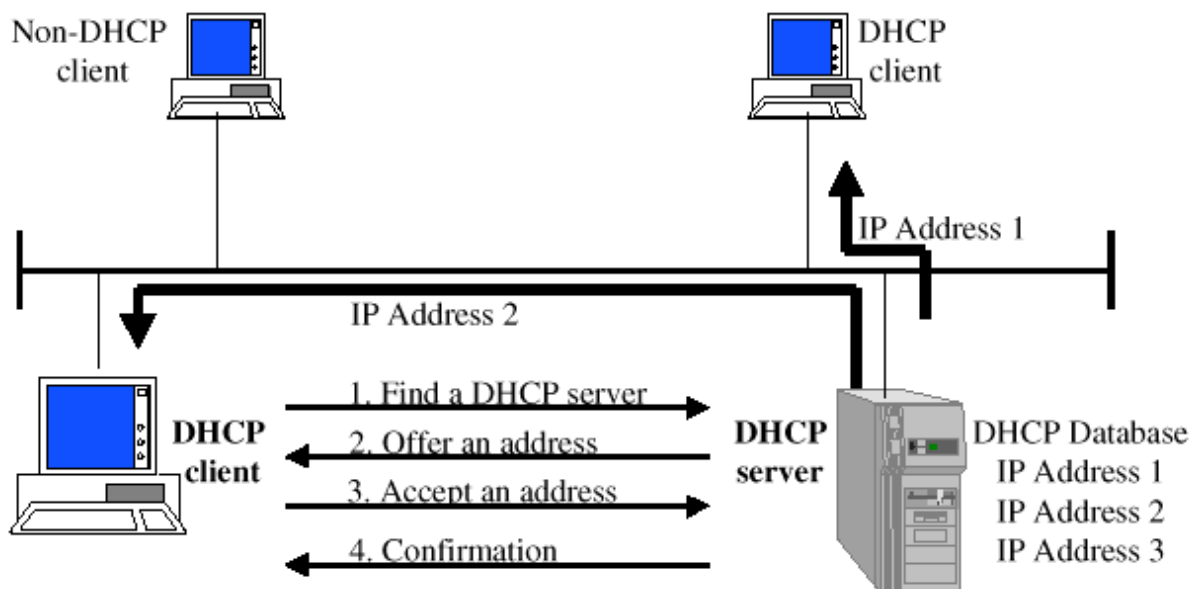
Each time a DHCP client starts, it requests IP an address from a DHCP server. When a DHCP server receives a request, it selects IP addressing information from a pool of addresses defined in its database and offers it to the DHCP client. If the client accepts the offer, the IP addressing information is leased to the client for a specified period of time.

In addition, the DHCP server will supply a subnet mask and optional values such as default gateway address, Domain Name Server (DNS) address and WINS (Windows Internet Name Service) address.

Non-DHCP clients still need to be configured manually with static addresses.

If there is no available IP addressing information in the pool to lease to a client, the client cannot initialise TCP/IP.

Dynamic Host Configuration Protocol - DHCP



DHCP supports three mechanisms for IP address allocation:

Manual Allocation

In this scheme, DHCP is simply used as a mechanism to deliver a predetermined network address and other configuration options to a host.

There is a one-to-one mapping between the unique client identifier (generally the Ethernet address) offered by the client during DHCP initialisation and the IP address returned to the client by the DHCP server. It is necessary for a network administrator to provide the unique client ID/IP address mapping used by the DHCP server.

Automatic Allocation

This is similar to manual allocation in that a permanent mapping exists between a host's unique client identifier and its IP address. However, in automatic allocation this mapping is created during the initial allocation of an IP address.

The IP addresses assigned during automatic allocation come from the same pool as dynamic addresses, but once assigned they cannot be returned to the free address pool without administrative intervention. Both automatic and manually assigned addresses are considered to have permanent leases.

Dynamic Allocation

DHCP assigns an IP address for a limited period of time. This IP address is known as a lease. This mechanism allows addresses that are no longer needed by their host to be automatically re-used.

DHCP uses a four-phase process to configure a DHCP client. In the first two phases the client requests a lease from a DHCP server and a

DHCP server offers an IP address to the client.

IP Lease Request

The first time a client is initialised, it requests an IP address lease by broadcasting a request to all DHCP servers. Because the client does not have an IP address or know the IP address of a DHCP server, it uses 0.0.0.0 as the source address and 255.255.255.255 as the destination address.

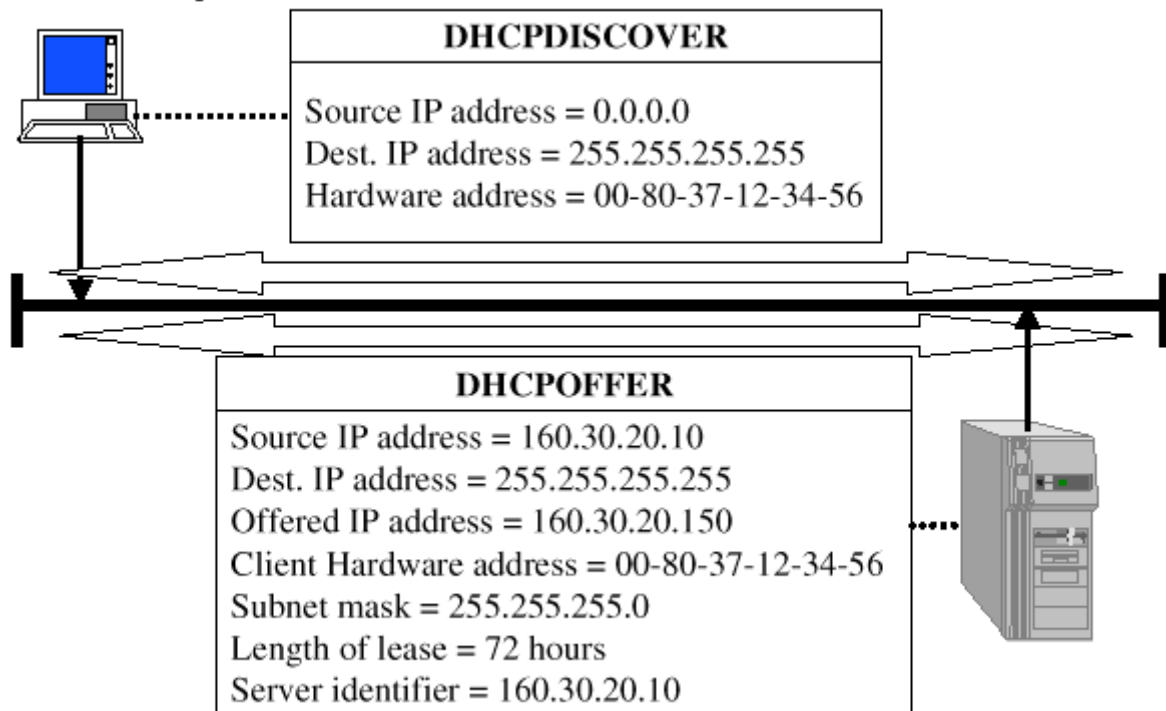
The request for a lease is sent in a DHCPDISCOVER message. This message also contains the client's hardware address and computer name, so that DHCP servers know which client sent the request.

The client sets a timer when sending a DHCPDISCOVER message. If it does not receive a response before the timer expires, it will resend the DHCPDISCOVER message.

IP Lease Offer

All DHCP servers that receive the request, and have a valid configuration for the client, broadcast an offer with the following information: the client's hardware address, an offered IP address, a subnet mask, the length of the lease and a server identifier (the IP address of the offering DHCP server). A broadcast is used because the client does not yet have an IP address. The offer is sent as a **DHCPOFFER** message. The DHCP server reserves the IP address so that it will not be offered to another DHCP client.

DHCP Operation



IP Lease Selection

The DHCP client selects the IP address from the first offer it receives. After the client receives an offer from at least one DHCP server, it broadcasts to all

DHCP servers that it has made a selection by accepting an offer. The broadcast is sent in a **DHCPREQUEST** message and includes the identifier (IP address) of the server whose offer was accepted. All other DHCP servers then retract their offer so that their IP addresses are available for the next IP lease request.

IP Lease Acknowledgement (Successful)

The DHCP server with the accepted offer broadcasts a successful acknowledgement to the client in the form of a DHCPACK message. This message contains a valid lease for an IP address and possibly other configuration information. When the DHCP client receives the acknowledgement, TCP/IP is completely initialised and is considered a bound DHCP client. Once bound, the client can use TCP/IP to communicate on the internetwork. The client stores the IP address, subnet mask and other IP addressing information locally.

IP Lease Acknowledgement (Unsuccessful)

An unsuccessful acknowledgement (DHCPNACK) is broadcast if:

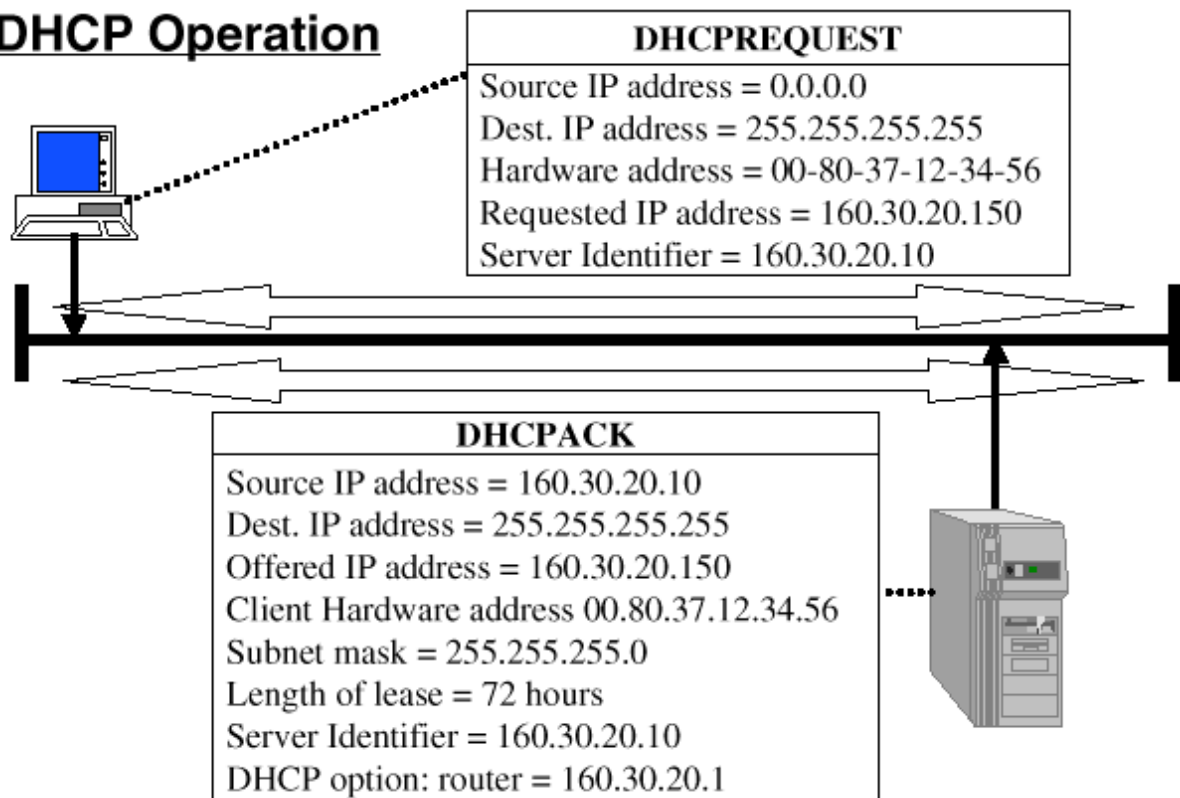
- The client is trying to lease its previous IP address and the IP address is no longer available
- The IP address is invalid because the client has been physically moved to a different subnet.

When the client receives an unsuccessful acknowledgement, it returns to the process of requesting an IP lease.

IP Lease Renewal

All DHCP clients attempt to renew their lease when 50 percent of the lease time has expired. To renew its lease, a DHCP client sends a DHCPREQUEST message directly to the DHCP server from which it obtained the lease. If a lease cannot be renewed by the original DHCP server, the client still uses the address as 50 percent of the lease life is still available. The client will attempt to contact any available DHCP server when 87.5 percent of the lease time has expired. If this is unsuccessful and the lease expires, the DHCP client can no longer use the IP address and communication over TCP/IP stops until a new IP address can be assigned to the client.

DHCP Operation



DHCP Interaction Through Routers

Routers can be configured to act as 'relay agents' in order to allow DHCP servers located on one IP network to serve configuration requests from remote networks.

A relay agent that conforms to RFC 1542 relays DHCP packets to a remote network even though they are broadcast packets. Before relaying a DHCP message from a DHCP client, the agent examines the gateway IP address field. If the field has an IP address of 0.0.0.0 the agent fills it with the router's IP address.

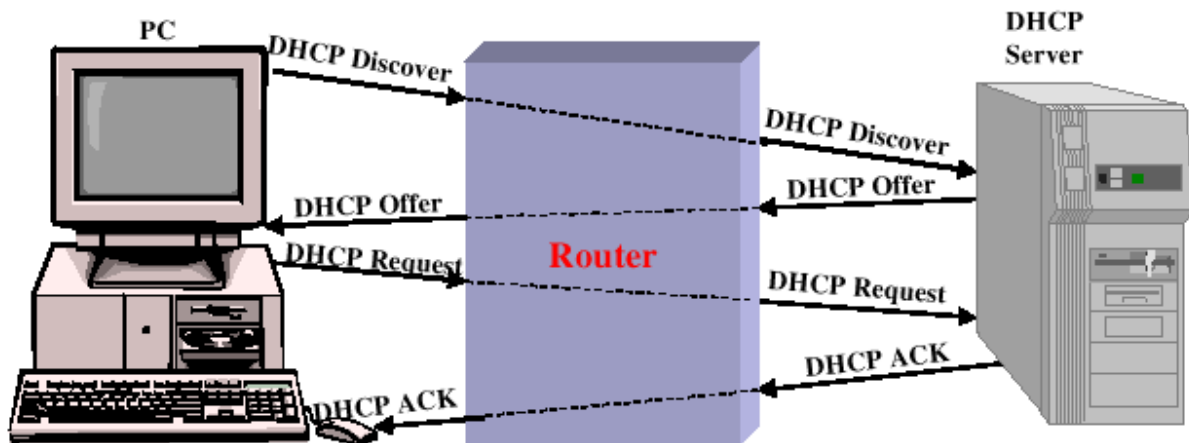
When the DHCP server receives the message it examines the relay IP address field in order to see if it has a DHCP scope (a pool of IP addresses) that can be used to supply an IP address lease. If the DHCP server has multiple scopes the address in the relay IP address field identifies the DHCP scope from which to offer an IP address lease. This process allows one DHCP server to manage different scopes for different networks.

When it receives the DHCP Discover message, the DHCP server sends a

DHCP Offer directly to the relay agent identified in the gateway IP address field, and the agent relays the message to the client. The client's IP address is unknown, thus it has to be broadcast on the local subnet.

Similarly a DHCP request message is relayed from client to server and a DHCP ACK message is relayed from server to client according to RFC 1542.

DHCP interaction through routers



DHCP Message Format

Most of the fields (all except two) in a DHCP message are identical to fields in a BOOTP message.

DHCP interprets BOOTP's UNUSED field as a 16-bit FLAGS field. Only the high-order bit of the flags field has been assigned a meaning. A client can set this bit to request that the server respond using hardware broadcast instead of hardware unicast.

The DHCP OPTIONS field has the same format as the VENDOR SPECIFIC AREA in BOOTP, and DHCP honours all the vendor specific information items defined for BOOTP. As in BOOTP each option consists of a 1-octet code field and a 1-octet length field followed by octets of data which comprise the option.

The option used to define a DHCP message type consists of exactly three octets. The first octet contains the code 53, the second contains the length 1, and the third contains a value (listed below) used to identify one of the possible DHCP messages.

1. DHCPDISCOVER
2. DHCPOFFER
3. DHCPREQUEST
4. DHCPDECLINE
5. DHCPACK
6. DHCPNACK
7. DHCPRELEASE

DHCP Message Format

0	8	16	24	31
OP	HTYPE	HLEN	HOPS	
TRANSACTION ID				
SECONDS		FLAGS		
CLIENT IP ADDRESS				
YOUR IP ADDRESS				
SERVER IP ADDRESS				
ROUTER IP ADDRESS				
CLIENT HARDWARE ADDRESS (16 OCTETS)				
SERVER HOST NAME (64 OCTETS)				
BOOT FILE NAME (128 OCTETS)				
OPTIONS (VARIABLE)				

19. The Need for IPV6

If IPv4 works so well then why change?

The main cause for change was due to the limited address space. When IP was defined only a few computer networks existed. The designers decided to use 32-bit addresses which would allow them to include over a million networks. The global internet is, however, growing exponentially, with the size more than doubling annually. At this current rate, all prefixes will soon be assigned and no further growth will be possible.

Secondary reasons for change are new Internet applications. For example, applications that deliver audio and video need to deliver data at regular intervals. To keep such information flowing through the Internet without disruption, IP must avoid changing routes frequently. These new requirements led to the development of IPv6.

If required security can be implemented in IPv6. An authentication header guarantees that a packet is actually coming from the host indicated in its source address, unlike in IPv4 where the packet could be coming from a host other than that indicated in the source. This is known as “spoofing”.

IPv4 and IPv6

● If IPv4 works so well then why change?

- Dramatically increase the number of IP addresses
- Provide better support for real-time applications
- Security features

The new features in IPv6 can be grouped into the following categories:

19.1 Address Size

IPv6 uses 128-bit addresses instead of the 32-bit addresses of IPv4. This is an increase of address space by a factor of 2^{96} . The address space provided by

IPv6 is large enough to accommodate continued growth of the Internet for many decades. There are enough addresses supported by IPv6 to provide an order of 6×10^{23} unique addresses per square metre of the surface of the earth.

19.2 Improved Options Mechanism

IPv6 options are placed in separate optional headers that are located between the IPv6 header and the transport layer header. Most of these optional headers are not examined or processed by any router on the packet's path. This simplifies and speeds up router processing of IPv6 packets compared to IPv4 packets.

19.3 Address Auto-configuration

This capability provides for dynamic assignment of IPv6 addresses via stateful or stateless address autoconfiguration. DHCP is termed a stateful address configuration tool because it maintains static tables that determine which addresses are assigned to a new or moved stations. A version of DHCP has been developed for IPv6. IPv6 also supports a stateless address autoconfiguration service that does not require a manually configured server.

Stateless autoconfiguration makes it possible for devices to configure their own addresses with the help of a local IPv6 router. Typically the device combines its 48-bit MAC address with a network prefix it learns from a neighbouring router.

19.4 Increased Addressing Flexibility

IPv6 includes the concept of an anycast address, for which a packet is delivered to just one of a set of nodes. The scalability of multicast routing is improved by adding a scope field to multicast addresses.

19.5 Support for Resource Allocation

instead of the type of service field in IPv4, IPv6 enables the labelling of packets belonging to a particular traffic flow for which the sender requests special handling. This aids in the support of specialised traffic, such as real-time video.

19.6 Security Capabilities

IPv6 includes features that support authentication and privacy.

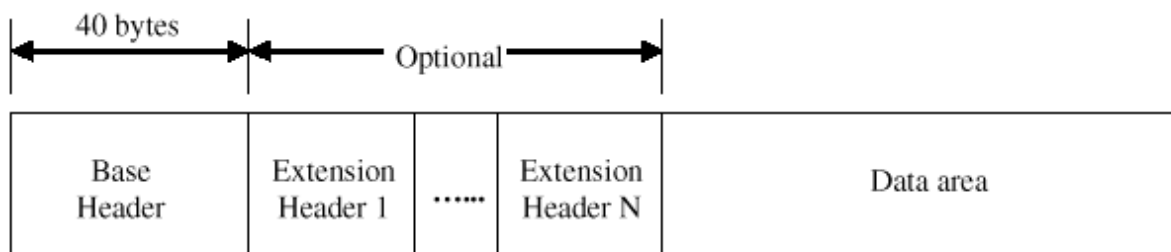
New features of IPv6

- ◆ **Address size**
 - 128-bit addresses
- ◆ **Improved option mechanism**
 - simplifies and speeds up router processing of IPv6 packets
- ◆ **Address autoconfiguration**
 - dynamic assignment of IPv6 addresses
- ◆ **Increased addressing flexibility**
 - anycast address
- ◆ **Support for resource allocation**
 - labelling of packets to handle specialised traffic
- ◆ **Security capabilities**
 - authentication and privacy

19.7 The IPv6 Packet Format

The IPv6 datagram begins with a base header, which is followed by zero or more extension headers, followed by data. The only header required is that of the IPv6 header. This is of fixed size with a length of 40 octets compared to 20 octets for the mandatory portion of the IPv4 header.

The IPv6 Packet Format



The IPv4 Header

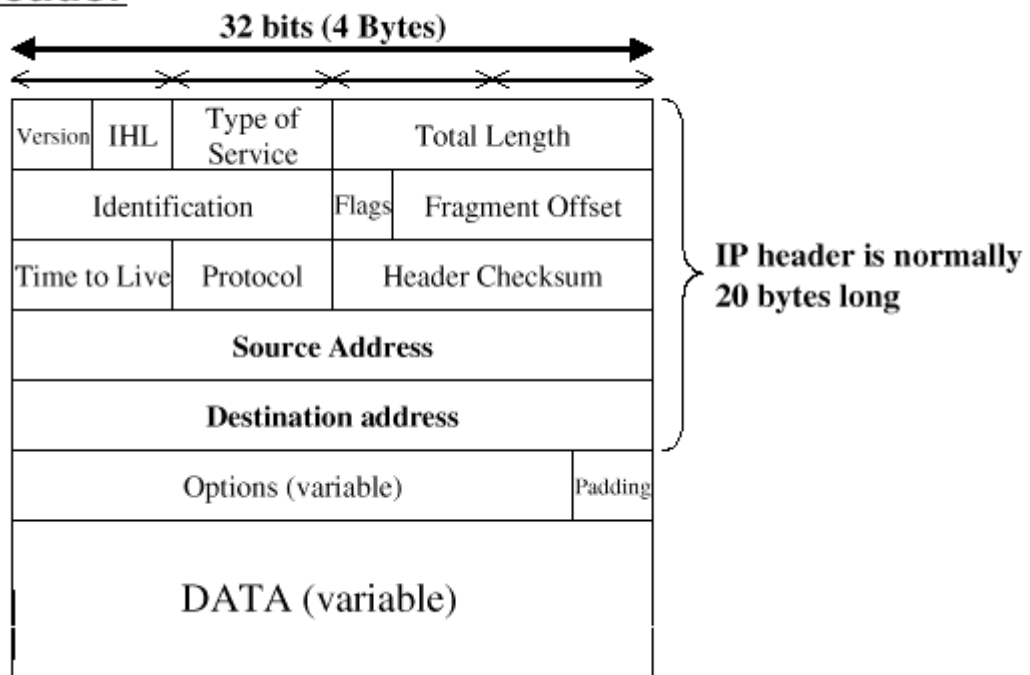
The IPv4 header has the following characteristics:

- The IPv4 header contains 14 fields (including options and padding), whereas IPv6 only requires 8 fields.
- Both headers carry version numbers and source/destination addresses.
- In IPv6 the following IPv4 fields are not included in the base header:

Internet header length, type-of-service, identification, flags, fragment offset and header checksum.

- The Internet header length is no longer required due to the fixed header length of all IPv6 packets.
- The functionality of the IPv4 type-of-service field has been transferred to the two new IPv6 fields: flow label and priority.
- The IPv4 fragmentation fields have been made optional headers in IPv6.
- The IPv4 checksum fields have been abandoned in IPv6, due to the prevalence of error checking at other levels of the protocol stack.
- The total length of the IPv4 packet has been retained in the guise of the IPv6 payload length field. This field does not, however, include the length of the IPv6 header. This is always assumed to be 40 bytes.
- The Time-To-Live (TTL) field of IPv4 has been changed to the IPv6 hop limit field. Although the names are different, both fields are used by routers to decrement a maximum hop value by 1 for each hop of the end-to-end route.

IPv4 Header

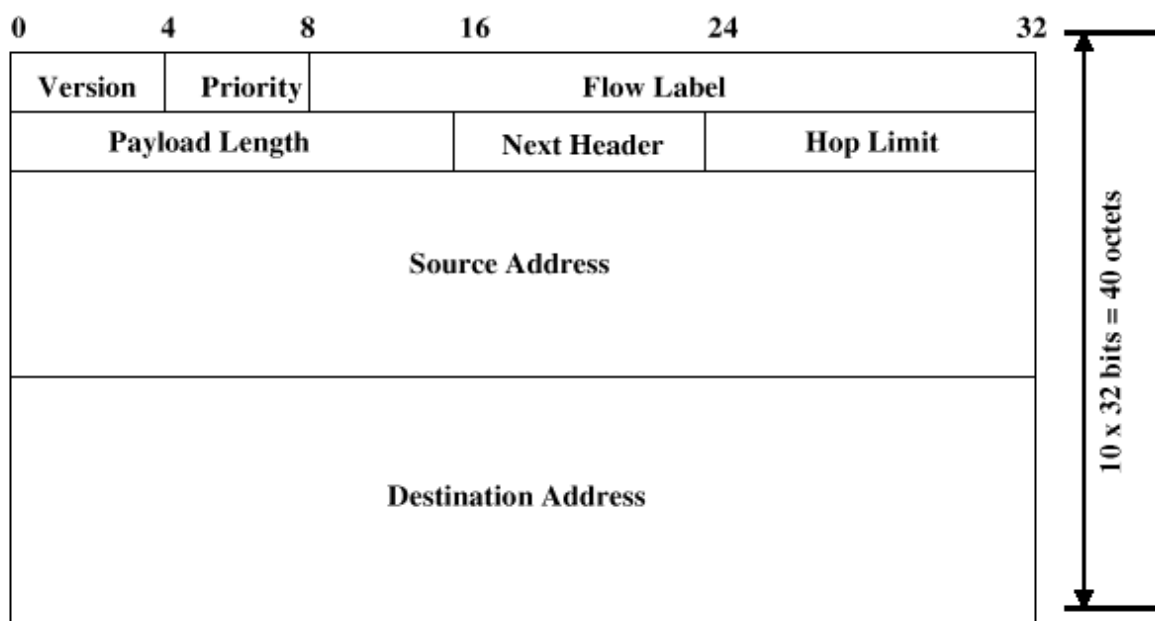


The IPv6

The IPv6 base header has a fixed length of 40 octets, consisting of eight fields:

Packet Element	Number of Bits	Purpose
Version	4	IP version number; the value is 6.
Priority	4	Priority value of each packet specifies the traffic class. Values between 0 and 7 are defined for congestion controlled traffic (data) and between 8 and 15 for non-congestion controlled traffic (video and audio).
Flow Label	24	Used by applications that require a performance guarantee to specify the path. The IPv6 standard defines a flow as a sequence of packets sent from a particular source to a particular destination. A flow is uniquely identified by the combination of source address and a 24-bit flow label. Thus all packets that are to be part of the same flow are assigned the same flow label by the source.
Payload Length	16	Specifies the size of the data being carried.
Next Header	8	Identifies the type of header immediately following the IPv6 header, for example, a TCP/UDP header or a IPv6 optional header.
Hop Limit	8	The remaining number of hops for this packet. The hop limit is set to a desired maximum value by the source and decremented by 1 by each node that forwards this packet. The packet is discarded if the hop limit is decremented to zero.
Source Address	128	Address of the sender of the packet.
Destination Address	128	Address of the intended recipient of packet.

IPv6 Base Header



There are seven kinds of extension header:

Extension Header	Description
Hop-by-hop Options Header	Defines special options that require hop-by-hop processing
Destination Option Header -1	Contains optional information to be examined by the first destination listed in the IPv6 address field. This header can also be read by a subsequent destination listed in the source routing header address fields.
Routing Header	Allows a source node to specify a list of IP addresses that dictate what path a packet will traverse.
Fragment Header	Contains fragmentation and reassemble information.
Authentication Header	Provides packet integrity and authentication.
Encapsulated Security Payload Header	Provides privacy.
Destination Options Header -2	Contains optional information to be examined only by the final destination node.

These headers are supplied to provide extra information, but are encoded in an efficient way. Processing speed is increased because each header is a fixed length. Each one of the seven extension headers is optional and if more than one is present, they must appear directly after the fixed header and in the preferred order.

The IPv6 header and extension header contain a Next Header Field. This field identifies the type of the header immediately following. If the next header is an extension, then this field contains the type identifier of that header, otherwise this field contains the protocol identifier of the upper layer protocol using IPv6.

IPv6 Extension Header

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options -1	Information for 1 st destination
Routing	Full or partial route to follow
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents
Destination options -2	Additional information for the final destination only

Hop-by-hop Options Header

The hop-by-hop option header carries optional information that, if present, must be examined by every router along the path. This header consists of the following:

- Next Header (8 bits): identifies the type of header immediately following this header.
- Header Extension Length (8 bits): Length of this header in 64-bit units, not including the first 64 bits.
- Options: a variable length field consisting of one or more option definitions. Each definition is in the form of three subfields:
 - Option type (8bits), which identifies the option
 - Length (8 bits), which specifies length of the option data field in octets
 - Option data, which is a variable-length specification of the option

Destination Options Headers

There are two variations of this header, each with a different position in the packet. The first variation is for carrying information to the first destination listed in the IPv6 address field. This header can also be read by a subsequent destination listed in the source routing header address fields. The second

variation is used for optional information that is only to be read by the final destination.

For efficiency, the first variation is typically located towards the front of the header chain, directly after the hop-by-hop header (if any). The second variation is relegated to a position at the end of the extension header chain, which is typically the last IPv6 optional header before transport and payload.

Hop-by-hop Options & Destination Options Headers

● **Hop-by-hop Options Header**

- Read by all routers along the path
- useful for transmitting management information or debugging commands to routers

● **Destination Options Header**

- 2 types
 - one for 1st destination
 - one for final destination

Routing Header

The routing header allows a source node to specify a list of IP addresses that dictate what path a packet will traverse.

RFC 1883 defines a version of this routing header called 'Type 0', which gives a sending node a great deal of control over the route of each packet. Type zero routing headers contain a 24-bit field that indicates how intermediate nodes may forward a packet to the next address in the routing header. Each bit in the 24-bit field indicates whether the next corresponding destination address must be a neighbour of the preceding address (1=strict, must be a neighbour; 0=loose, need not be a neighbour).

When routing headers are used for 'strict' forwarding, a packet visits only routers listed in the routing header. For example, if routers 2 and 3 are listed as strict but are not adjacent to each other (that is, in order to get from 2 to 3 a packet must pass through some other router) packets will be dropped at 2. This is a valuable feature when security and traffic control require that packets take a rigidly defined path. In 'loose' forwarding, unlisted routers can be visited by a packet.

When Type 0 routing headers are used, the initial IPv6 header contains the destination address of the first router in the source route and not the final destination address.

Routing Header

- Specifies a list of IP addresses that dictate what path a packet will traverse
- Type zero routing headers indicate how intermediate nodes may forward a packet to the next address in the routing header
 - strict forwarding, packets only visit routers listed in the routing header
 - loose forwarding, unlisted routers can be visited by a packet

If a router does not recognise the routing type value, it must discard the packet. Type 0 routing has been defined and has the fields outlined in the following table:

Table 3.7 IPv6 Routing Header

Field	Bits	Purpose
Next Header	8	Identifies the type of header immediately following this header.
Routing Type	8	Currently set to zero
Num Addrs	8	Number of addresses in the routing header. The maximum value is 24.
Next Addr	8	Index of the next address to be processed. Is initialised to zero by the originating node, and is incremented as each address is visited.
Reserved	8	For future use
Strict/Loose Bit Mask	24	Numbered from left to right, with each bit corresponding to one of the addresses. Each bit indicates whether the corresponding next destination address must be a neighbour of the preceding address (1 = strict, must be a neighbour; 0 = loose, need not be a neighbour).

Routing Header Format

Next Header	Type	Number of Addresses	Next address
Reserved	Strict/loose Bit Map		
1 - 24 Addresses			

Fragment Header

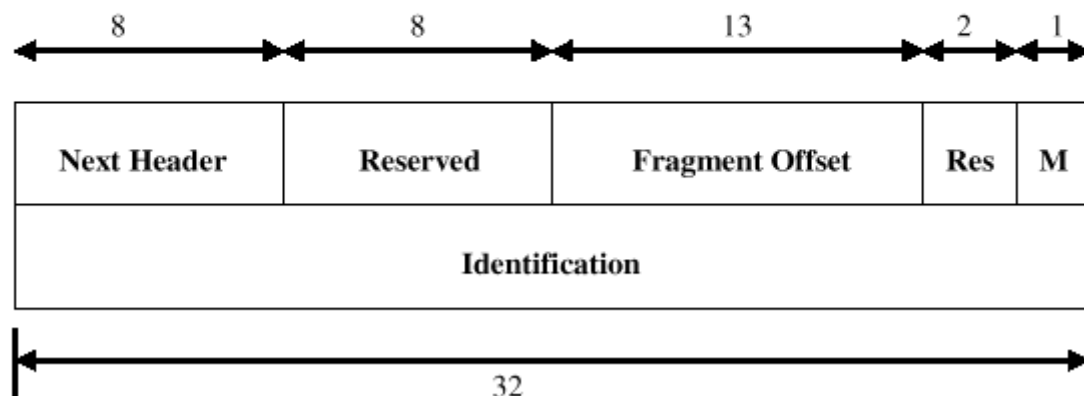
In IPv6, fragmentation may only be performed by source nodes, not by routers along a packet's delivery path. End nodes performing fragmentation can determine the smallest MTU of a path using the MTU path discovery process.

The source node sends out a packet with an MTU as large as the local interface can handle. If this MTU is too large for some link along the path, an ICMP 'Datagram too big' message will be sent back to the source. This message will contain a 'Datagram too big' indicator and the MTU of the affected link. The source can then adjust the packet size downward and retransmit another packet. This process is repeated until a packet gets all the way to the destination node. The discovered MTU is then used for fragmentation purposes.

The table below outlines the elements of the Fragment Header:

Field	Bits	Purpose
Next Header	8	Identifies the type of header immediately following this header.
Reserved	8	For future use.
Fragment Offset	13	indicates where in the original packet the payload of this fragment belongs. It is measured in 64-bit units; which implies that fragments must contain a data field that is a multiple of 64 bits.
Res	2	Reserved for future use.
M Flag	1	1 = more fragments; 0 = last fragment.
Identification	32	Intended to uniquely identify the original packet.

Fragment Header



Authentication Header

The IPv6 authentication extension header gives network applications a guarantee that the packet did in fact come from an authentic source as indicated in its source address. This authentication is particularly important to safeguard against intruders who configure a host to generate packets with forged source addresses. This type of source address masquerading can spoof a server so that access may be gained to valuable data, passwords or network control utilities.

With IPv6 authentication headers, hosts establish a standards-based security

association that is based on the exchange of algorithm-independent security keys (for example MD5).

In a client/server session, for instance, both the client and the server need to know the key. Before each packet is sent, IPv6 authentication creates a checksum based on the key combined with the entire contents of the packet.

This checksum is then re-run on the receiving side and compared. This approach provides authentication of the sender and guarantees that data within the packet has not been modified by an intervening party. Authentication can take place between clients and servers or clients and clients on the corporate backbone. It can also be deployed between remote stations and corporate dial-in servers to ensure that the perimeter of the corporate security is not breached.

Authentication Header

- ◆ **The authentication header provides authentication and integrity**
- ◆ **the authentication header extension to IPv6 ensures that a packet is actually coming from the host indicated in its source address**

Encryption Security Payload Header

Authentication headers eliminate a number of host spoofing and packet modification hacks. They do not, however, prevent nondisruptive reading, such as sniffing (reading network traffic) of data as it traverses the Internet and corporate backbone networks. This area is dealt with by the Encapsulating

Security Payload (ESP) service of IPv6 which provides integrity and confidentiality in IPv6 datagrams. ESP provides encryption at the network layer, making it available to all applications in a highly standardised fashion.

IPv6 ESP is used to encrypt the transport-layer header and payload (for example, TCP or UDP) or the entire IP datagram. Both these methods are accomplished with an ESP extension header that carries encryption parameters and keys end-to-end.

IPv6 has two modes to provide confidentiality:

Transport Mode ESP

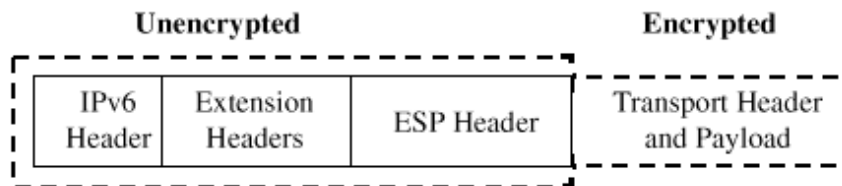
In this mode only the payload is encrypted. The IP header and IP options are unencrypted and are used for routing the packet. The receiver decrypts the ESP and continues to use the unencrypted header as an IP header if necessary.

Tunnel Mode ESP

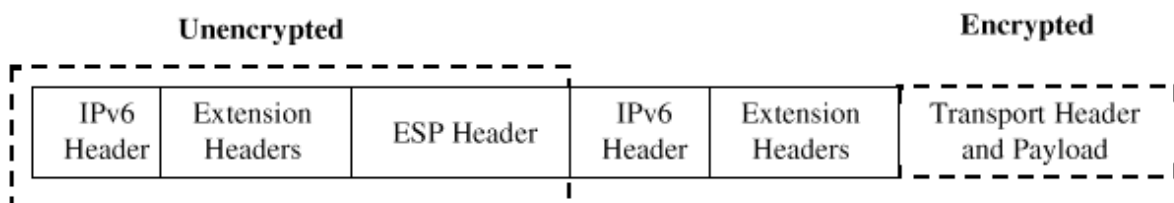
In this mode, the original IP datagram and header are encrypted. This entire ESP frame is placed within a new datagram with an unencrypted IP header. All additional unencrypted information such as routing header are placed between the IP header and the encapsulated security payload. The receiver strips off the cleartext IP header and decrypts the ESP.

ESP- Encrypted Security Payload

Transport Mode



Tunnel Mode



Like IPv4, IPv6 assigns a unique address for each connection between a computer and a physical network. Thus, if a computer connects to three physical networks the computer is assigned three addresses. IPv6 also separates each such address into a prefix that identifies the network and a suffix that identifies a particular computer on the network.

Although IPv6 adopts the same approach for assigning computer addresses, IPv6 addressing differs in several ways.

- First, all address details are completely different.
- Second, IPv6 defines a set of special addresses that differ dramatically from IPv4 special addresses. In particular, IPv6 does not include a special address for broadcasting on a given network. Instead it uses a form of multicasting.

There are three types of IPv6 addresses:

- Unicast: the address corresponds to a single computer. A datagram sent to the address is routed along the shortest path to the computer.
- Multicast: the address corresponds to a set of computers, possibly at many locations. Membership in the set can change at any time. When a datagram is sent to the address, IPv6 delivers one copy of the datagram to each member of the set.
- Anycast: the address corresponds to a set of computers that share a common address prefix. A datagram sent to the address is routed along the shortest path and delivered to one of the computers, typically the 'nearest' one according to current routing protocol metrics. This would, for example, allow an enterprise to use a single anycast address to forward

packets to a number of different routers on its ISP's backbone. If all of the providers' routers have the same anycast address, traffic from the enterprise will have several redundant access points to the Internet. If one of the backbone routers goes down, the next nearest device automatically will receive the traffic.

Writing 128-bit numbers can be confusing. For example, consider a 128-bit number written in dotted decimal notation:

105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

In order to help reduce the number of characters in an address, the designers of IPv6 propose using a more compact syntactic form known as hexadecimal notation in which each group of 16 bits is written in hexadecimal with a colon separating groups. For example, when the above number is written in colon hex, it becomes:

69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF

As illustrated in the example shown, colon hex notation requires fewer characters to express an address. An additional optimisation known as zero compression further reduces the size. Zero compression replaces sequences of zeroes with two colons. For example, the address:

FF0C:0:0:0:0:0:0:B1
can be written as:

FF0C::B1

Leading zeros within a group can be omitted, so 0123 can be written as 123. In order to help ease the transition to the new protocol, the designers mapped the existing IPv4 addresses into the IPv6 address space. Any IPv6 address that begins with 96 zero bits contains an IPv4 address in the low-order 32 bits.

IPv6 Colon Hexadecimal Notation

- **Consider a 128-bit number written in dotted decimal notation:**

- 105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255

- **This number written in hex notation**

- 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF

- **Leading zeros within a group can be omitted**

- **One or more groups of 16 zeros can be replaced by a pair of colons**

- for example: FF0C:0:0:0:0:0:0:B1 can be written as:

- FF0C::B1

20. Transition to IPv6

IPv6 hosts and routers will need to retain backward compatibility with IPv4 devices for an extended time period (possibly years or even indefinitely). In order to accomplish this goal, IPv6 transition relies on several special functions that have been built into the IPv6 standards work, including dual-stack routers and tunnelling IPv6 via IPv4.

Tunnelling In most organisations where IPv6 is deployed incrementally, there is a strong possibility that all IPv6 hosts will not have direct connectivity to each other via IPv6 routers. In many cases there will be islands of IPv6 topology surrounded by an ocean of IPv4. Fortunately, IPv6 designers have fashioned transition mechanisms that allow IPv6 hosts to communicate over intervening IPv4 networks.

The essential technique of these mechanisms is IPv6 over IPv4 tunnelling, which encapsulates IPv6 packets in IPv4 packets. Tunnelling allows IPv6 implementations to co-exist with IPv4 without any change to IPv4 components.

A dual-stack router or host on the 'edge' of the IPv6 topology simply appends an IPv4 header to each IPv6 packet and sends it as native IPv4 traffic through existing links. IPv4 routers forward this traffic without knowledge that IPv6 is involved. On the other side of the tunnel, another dual-stack router or host de-encapsulates the IPv6 packet and routes it to the ultimate destination using standard IPv6 protocols.

In order to facilitate different administrative needs, IPv6 transition mechanisms include two types of tunnelling: automatic and configured.

To build configured tunnels, administrators manually define IPv6-to-IPv4 address mappings at the tunnel endpoints.

Automatic tunnels use 'IPv4-compatible' addresses, which are hybrid IPv4/IPv6

addresses. Compatible addresses are created by adding leading zeros to the 32-bit IPv4 address to pad them out to 128 bits. When traffic is forwarded with compatible addresses, the device at the tunnel entry point can automatically address encapsulated traffic by simply converting the IPv4-compatible 128-bit address to a 32-bit IPv4 address. On the other side of the tunnel, the IPv4 header is removed to reveal the original IPv6 address. Automatic tunnelling allows IPv6 hosts to dynamically exploit IPv4 networks, but it does require the use of IPv4-compatible addresses, which do not bring the benefits of the 128-bit address space.

IPv6 nodes using IPv4-compatible addresses cannot take advantage of the extended address space, but they can exploit the other IPv6 enhancements (including flow labels, authentication, encryption, multicast and anycast). Once a node is migrated to IPv6 with IPv4-compatible addressing, the door is open for a fairly painless move to the full IPv6 address space.

Transition to IPv6

● Tunnelling

– Configured

- manually configuration of IPv6/IPv4 mappings
- whole IPv6 address space can be used

– Automatic

- compatible address space
- no advantage of the extended address space

21. Remote Command Execution

Telnet

The TCP/IP protocol suite includes a simple remote terminal protocol called Telnet. Telnet allows a user at one site to establish a TCP connection to a login server at another. Telnet then passes the keystrokes from the user's keyboard directly to the remote computer, as if they had been typed on a keyboard attached to the remote computer. Telnet also carries output from the remote computer back to the user's screen.

Telnet client software allows the user to specify the remote computer, either by giving its domain name or IP address.

In the login procedure, the username and password are transferred unsecured (plain text) through the network.

Telnet offers a number of basic services. It defines a network virtual terminal (NVT) that provides a standard interface to remote systems. Client programs do not have to understand the details of all possible remote systems, as they are built to use the standard interface. Telnet includes a mechanism that allows the client and server to negotiate options. Telnet provides a set of standard options. For example, one of the options controls whether data passed across the connection uses the standard 7-bit ASCII character set or an 8-bit character set. Also, Telnet treats both ends of the connection symmetrically. In particular,

Telnet does not force client input to come from a keyboard, nor does it force the client to display output on a screen. Thus Telnet allows an arbitrary program to become a client. Furthermore, either end can negotiate options.

Network Virtual Terminal

The characteristics of a locally attached terminal are known to the application, but the characteristics of a remote terminal are likely be unknown to the application. Hence, when Telnet is used, the client converts the terminal characteristics of the user to those of a universal terminal type, called virtual terminal. The server converts the virtual terminal characteristics to make them appear as though generated by a local terminal. This feature is called network virtual terminal (NVT).

The NVT represents the lowest common denominator of existing terminal features:

- A bi-directional character device with a printer and a keyboard
- Operates in scroll mode
- Unlimited line/page length
- Uses seven-bit ASCII characters encoded in eight-bit octets

Trivial File Transfer Protocol

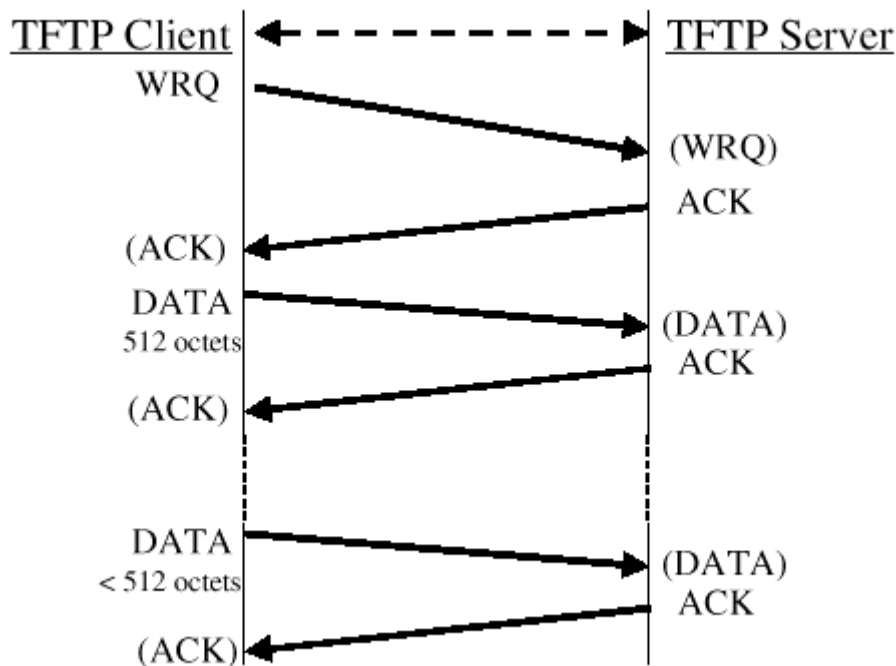
There exists a second file transfer service known as Trivial File Transfer Protocol (TFTP). TFTP is an extremely simple protocol used to transfer files.

TFTP differs from FTP in several ways:

- Communication between a TFTP client and server uses UDP (port 69) and not TCP. Therefore, TFTP lacks the FTP/TCP features such as sequencing, windowing and full duplex operation.
- TFTP only supports file transfer. That is, TFTP does not support interaction and does not have a large set of commands.
- TFTP does not have authorisation. A client does not send a login name or password. Therefore, a file can be transferred only if the file permissions allow global access.

Although TFTP is not as powerful as FTP, it was designed with UDP to make it simple and small. Implementations of TFTP (and its required UDP, IP and device driver) can fit in read-only memory. This makes TFTP suitable for bootstrapping diskless systems, for example, X-terminals.

- TFTP is an extremely simple protocol to transfer files
- Communication between a TFTP client and server uses UDP (port 69) not TCP
- TFTP does not have authorisation
- TFTP always sends 512-byte blocks of data



TFTP Message Format

The first two bytes of the TFTP message are an opcode. For an RRQ and a WRQ the filename specifies the file on the server that the client wants to read from or write to. This filename is terminated by an octet of zero. The mode is one of the ASCII strings netascii or octet, again terminated by an octet of 0.

Netascii means the data are lines of ASCII text with each line terminated by the two-character sequence of a carriage return followed by a line feed (CR/LF).

Both ends must convert between this format and whatever the local host uses as a line delimiter. An octet transfer treats the data as 8-bit bytes with no interpretation.

Each data packet contains a block number that is later used in an acknowledgement packet. When reading a file the client sends an RRQ specifying the filename and mode. If the file can be read by the client, the server responds with a data packet with a block number of one. The client sends an ACK of block number one. The server responds with the next data packet with a block number of two. The client sends an ack of block number two. This continues until the file is transferred.

In the case of a WRQ, the client specifies the filename and mode. If the file can be written by the client, the server responds with an ACK of block number zero.

The client then sends the first 512 bytes of file with a block number of one. The server responds with an ACK of block number one.

The final TFTP message type is the error message, with an opcode of five. This is what the server responds with if an RRQ or WRQ can't be processed. Read and write errors during file transmission also cause this message to be sent, and transmission is then terminated. The error number gives a numeric error code, followed by an ASCII error message that might contain additional, operating system-specific information.

Format of TFTP Messages

2 octets	n octets	1 octet	n octets	1 octet
Read request (1)	Filename	0	Mode	0

2 octets	n octets	1 octet	n octets	1 octet
Write request (2)	Filename	0	Mode	0

2 octets	2 octets	Up to 512 octets
Data (3)	Block number	Data

2 octets	2 octets
Ack (4)	Block number

2 octets	2 octets	n octets	
Error (5)	Error code	Error message	0

Remote Procedure Call (RPC)

An RPC provides a programmer with the capability of having procedures (tasks, subroutines, and sub-procedures) executed remotely. The programmer declares to the compiler that a particular procedure is remote. The compiler inserts the proper code for an RPC to the client. On the server side, the procedure is declared to the compiler as the remote procedure. When the RPC is made, the client obtains control and forms a message to the server that contains the arguments for the remote procedure. The server unpacks the arguments, performs the procedure, packs the results of the procedure in a message, and returns it to the client. The client unpacks the message and returns control to the calling program. Having the compiler insert the RPC and the client/server provide results to the calling program shields the programmer from the complexities of the network.

RPC is a session layer protocol than can use either UDP or TCP transport. When RPC uses the transport layer UDP, the calling application program must be aware of this in order to perform its own timeout, retransmission and duplicate detection as the RPC does not provide these services. When TCP transport is used, RPC inhibits any retransmission since TCP assures a reliable safe delivery.

The session layer header for RPC contains:

- The transaction ID
- The identifier for the direction of transfer
- The protocol version number
- The program number
- The procedure number
- The authorisation (both information and verification)

From this information, the server is able to select and execute the proper

procedure and return the results to the client.

Remote Procedure Call

- RPC executes procedures (tasks, subroutines, and sub-procedures) remotely.
 - The client obtains control and forms a message to the server that contains the arguments for the remote procedure
 - The server unpacks the arguments, performs the procedure, packs the results of the procedure in a message, and returns it to the client
 - The client unpacks the message and returns control to the calling program
- RPC is a session layer protocol than can use either UDP or TCP transport.
- header contains the transaction ID, the identifier for the direction of transfer, the protocol version number, the program number, the procedure number, and the authorisation.

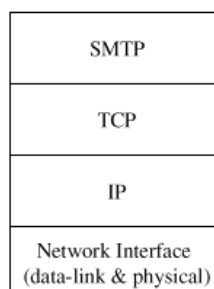
Simple Mail Transfer Protocol (SMTP)

The function of a mail program is to facilitate the exchange of electronic messages between users on a network. The Internet uses SMTP in order to provide a mail service. SMTP is an applications layer program that interfaces directly with the transport layer TCP. The destination, well-known port number 25, is used in the TCP header, which causes TCP to direct incoming segments to SMTP for processing.

In this section the components involved in the SMTP process and the commands and replies used by SMTP are examined. The operation of the SMTP protocol is also illustrated with an example.

Simple Mail Transfer Protocol (SMTP)

- SMTP is the Internet standard mail service
- Uses TCP port 25



SMTP Client Functions

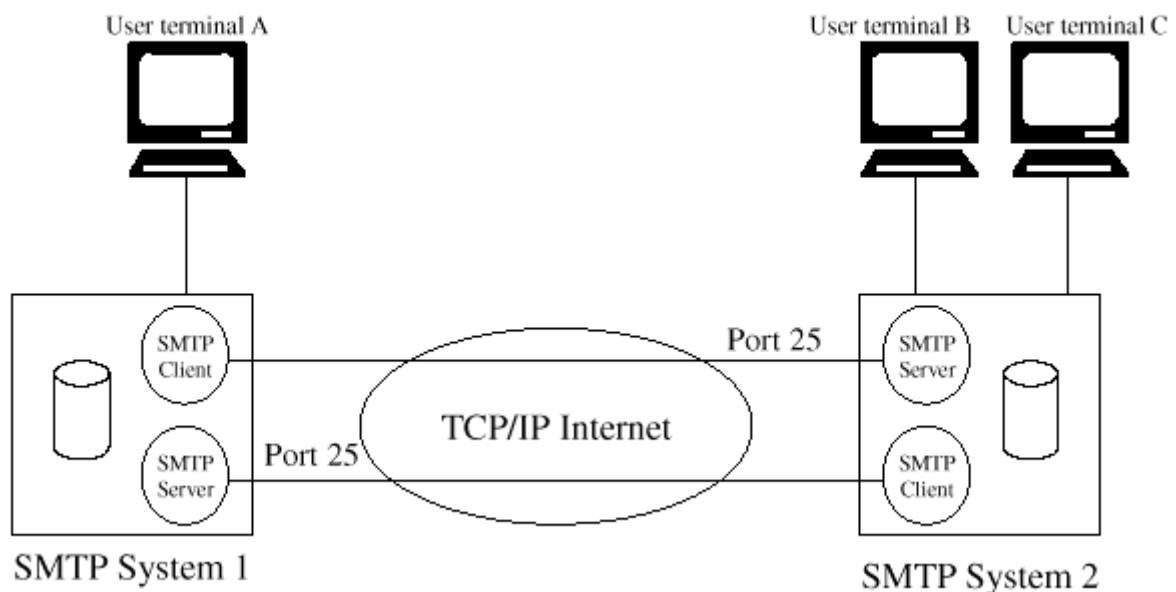
The SMTP client reads the next message from the queue of user agent inputs.

If an address in the standardised list is for this SMTP system, it is handed off directly to the SMTP server. Otherwise a TCP connection is established with each remote SMTP server machine indicated by the addresses in the standardised list. The SMTP client sends each address in the standardised list, one at a time, to the SMTP server associated with the address. The SMTP server sends an OK reply for each address received and the SMTP client marks the address in the standardised list as sent. When all addresses in the standardised list are sent, the SMTP sends a single copy of the message to each SMTP server with a TCP connection. The SMTP server sends a positive acknowledgement for the message, and the SMTP client is no longer responsible for the message.

SMTP Server Functions

The SMTP server constructs a header for each address received and places it in the queue of the appropriate mailbox, or an output queue of another SMTP server if the message is being forwarded. The header contains a pointer to the user's text and typically five lines of header constructed by the SMTP

SMTP Process (Contd)



Network File System

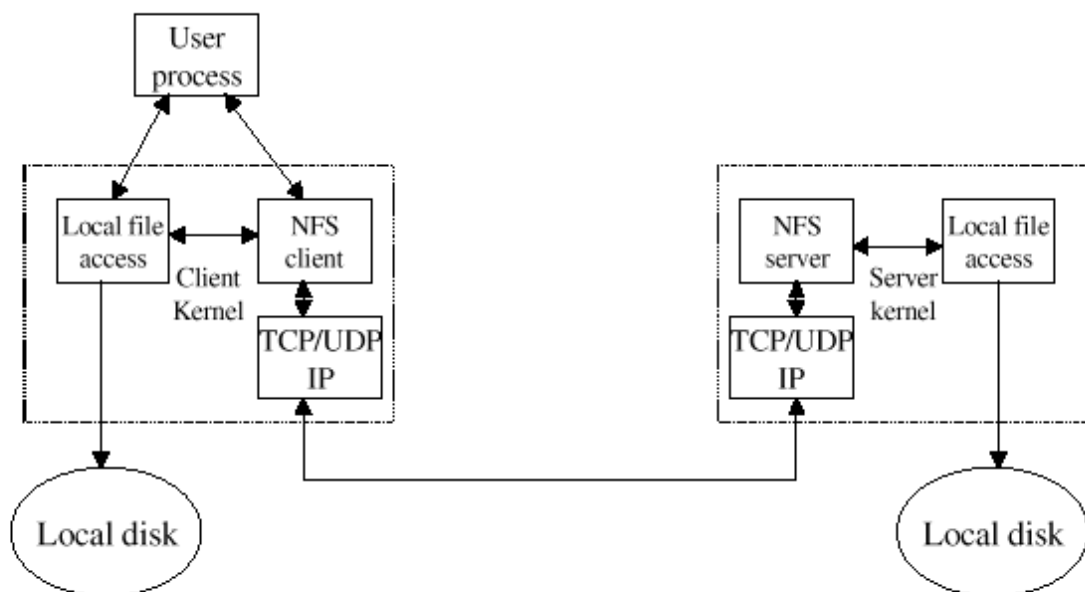
NFS provides transparent file access for clients to files and file systems on a server. This differs from FTP, which provides file transfer. With FTP a complete copy of the file is made. NFS accesses only the portion of a file that a process references, and a goal of NFS is to make this access transparent. This means that any client application that works with a local file should work with an NFS file, without any program changes whatsoever.

NFS clients access files on an NFS server by sending remote procedure call (RPC) requests to the server.

The diagram illustrates the typical arrangement of an NFS client and an NFS server. The following text describes the process:

1. It is transparent to the client whether it's accessing a local file or an NFS file. The kernel determines this when the file is opened. After the file is opened, the kernel passes all references to local files to the box labelled "local file access", and all references to an NFS file are passed to the "NFS client" box.
2. The NFS client sends RPC request to the NFS server through its TCP/IP module. NFS is used predominantly with UDP, but newer implementations can also use TCP.
3. The NFS server receives client requests as UDP packets on port 2049.
4. When the NFS server receives a client request, the requests are passed to its local file access routines, which access a local disk on the server.

Network File System



NFS provides transparent file access for clients to files and file systems on a server

NFS Procedures

The NFS server provides 15 procedures, described below.

GETATTR. Return the attributes of a file: type of file (for example, regular file, or directory), permissions, size of file, owner of file, last-access time, and so on.

SETATTR. Set the attributes of a file. Only a subset of the attributes can be set: permissions, owner, group owner, size, last-access time, and last modification time.

STAFS. Return the status of a file system: amount of available space, optimal size for transfer and so on.

LOOKUP. Lookup a file. This is the procedure called by the client each time a user process opens a file that's on an NFS server. A file handle is returned along with an attribute of the file.

READ. Read from a file. The client specifies the file handle, starting byte offset, and maximum number of bytes to read.

WRITE. Write to a file. The client specifies the file handle, starting byte offset, and number of bytes to write, and the data to write.

CREATE. Create a file.