Networking Concepts

COMP5311 Handbook by edwardtoday

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Mon, 12 Dec 2011 13:21:49 UTC

Contents

Articles

Communications protocol 1
Computer network 16
OSI model 31
TCP/IP model 39

Link Layer 48

Address Resolution Protocol 48
Asynchronous Transfer Mode 53
Bootstrap Protocol 59
Checksum 61
Circuit switching 63
Connection-oriented communication 65
Connectionless communication 66
Cyclic redundancy check 67
Data link layer 76
Error detection and correction 80
Ethernet 86
Ethernet frame 94
Ethernet hub 99
Frame (networking) 101
Frame Relay 101
Frame synchronization 106
Gateway (telecommunications) 108
Internet protocol suite 110
Label switching 116
LAN switching 117
Link layer 120
Logical link control 123
Media access control 125
Multiplexing 127
Network switch 132
Packet switching 137
Point-to-point protocol 141
Promiscuous mode 148

Protocol data unit 149
Proxy ARP 151
Quality of service 153
Reverse Address Resolution Protocol 162
Router (computing) 162
Self-synchronizing code 168
Shared medium 169
Spanning Tree Protocol 169
Tunneling protocol 179
Type of service 181
Virtual circuit 182

Internet Layer **185**

Autonomous system (Internet) 185
Best effort delivery 187
Broadcast address 187
Broadcasting (computing) 188
Classful network 190
Classless Inter-Domain Routing 193
End-to-end principle 199
Holddown 202
Hop-by-hop transport 203
Host (network) 204
ICMP Redirect Message 205
Internet Control Message Protocol 206
Internet Group Management Protocol 210
Internet Protocol 213
IP address 216
IP forwarding algorithm 223
IP fragmentation 224
IP in IP 225
IPv4 228
IPv4 subnetting reference 238
IPv6 240
IPv6 packet 252
Maximum transmission unit 259
Mobile IP 263
Multicast 266

Multicast address 269
Multihoming 274
Network address translation 277
Network interface controller 287
Network layer 289
Out-of-order delivery 291
Packet loss 291
Private network 293
Route poisoning 296
Subnetwork 297
Unicast 303

Transport Layer **305**

Additive increase/multiplicative decrease 305
Congestion window 306
Connection-oriented protocol 307
Connectionless protocol 308
Fast retransmit 308
Karn's Algorithm 309
Maximum segment size 310
Nagle's algorithm 311
Negative-acknowledge character 312
Network congestion 313
Path MTU Discovery 320
Port number 321
Random early detection 324
Retransmission (data networks) 326
Round-trip delay time 327
Silly window syndrome 328
Sliding window protocol 329
Slow-start 333
TCP congestion avoidance algorithm 335
Transmission Control Protocol 339
Transport layer 356
User Datagram Protocol 359

Routing Protocols in Different Layers **364**

Adaptive routing 364

Bellman–Ford algorithm 365
Border Gateway Protocol 368
Dijkstra's algorithm 381
Distance-vector routing protocol 387
Enhanced Interior Gateway Routing Protocol 391
Exterior Gateway Protocol 397
Flooding (computer networking) 397
Flooding algorithm 398
Hop count 399
Interior gateway protocol 400
Interior Gateway Routing Protocol 401
IS-IS 402
Link-state advertisement 404
Link-state routing protocol 412
Metrics (networking) 416
Multipath routing 417
Open Shortest Path First 419
Path vector protocol 432
Policy-based routing 433
Routing 434
Routing Information Protocol 439
Routing protocol 442
Routing table 444
Split horizon route advertisement 446

Application Layer **448**

Domain Name System 448
Dynamic Host Configuration Protocol 461
Hypertext Transfer Protocol 470

References

Article Sources and Contributors 477
Image Sources, Licenses and Contributors 491

Article Licenses

License 493

Communications protocol

A **communications protocol** is a system of digital message formats and rules for exchanging those messages in or between computing systems and in telecommunications. A protocol may have a formal description.

Protocols may include signaling, authentication and error detection and correction capabilities.

A protocol definition defines the syntax, semantics, and synchronization of communication; the specified behavior is typically independent of how it is to be implemented. A protocol can therefore be implemented as hardware or software or both.

Introduction

While there is no generally accepted formal definition of "protocol" in computer science, an informal definition, based on the previous, could be "a set of procedures to be followed when communicating". In computer science the word *algorithm* is a synonym for the word *procedure* so a protocol is to communications what an algorithm is to mathematics.

Communicating systems use well-defined formats for exchanging messages. Each message has an exact meaning intended to provoke a defined response of the receiver. A protocol therefore describes the *syntax*, *semantics*, and *synchronization* of communication. A *programming language* describes the same for computations, so there is a close analogy between protocols and programming languages: *protocols are to communications what programming languages are to computations*.[1]

The communications protocols in use on the Internet are designed to function in very complex and diverse settings. To ease design, communications protocols are structured using a layering scheme as a basis. Instead of using a single universal protocol to handle all transmission tasks, a set of cooperating protocols fitting the layering scheme is used.[2]

The layering scheme in use on the Internet is called the TCP/IP model. The actual protocols are collectively called the Internet protocol suite. The group responsible for this design is called the Internet Engineering Task Force (*IETF*).

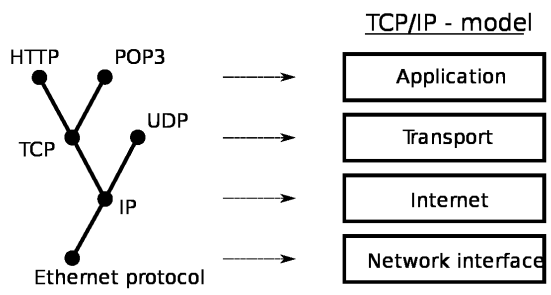


Figure 2. The TCP/IP model or Internet layering scheme and its relation to some common protocols.

Obviously the number of layers of a layering scheme and the way the layers are defined can have a drastic impact on the protocols involved. This is where the analogies come into play for the TCP/IP model, because the designers of TCP/IP employed the same techniques used to conquer the complexity of programming language *compilers* (design by analogy) in the *implementation* of its protocols and its layering scheme.[3]

Communications protocols have to be agreed upon by the parties involved. To reach agreement a protocol is developed into a *technical standard*.

Communicating systems

The information exchanged between devices on a network or other communications medium is governed by rules or conventions that can be set out in a technical specification called a communication protocol standard. The nature of the communication, the actual data exchanged and any state-dependent behaviors are defined by the specification.

In digital computing systems, the rules can be expressed by algorithms and data structures. Expressing the algorithms in a portable programming language, makes the protocol software operating system independent.

Operating systems are usually conceived of as consisting of a set of cooperating processes that manipulate a shared store (on the system itself) to communicate with each other. This communication is governed by well-understood protocols. These protocols can be embedded in the process code itself as small additional code fragments.[4] [5]

In contrast, communicating systems have to communicate with each other using shared transmission media, because there is no common memory. Transmission is not necessarily reliable and can involve different hardware and operating systems on different systems.

To implement a networking protocol, the protocol software modules are interfaced with a framework implemented on the machine's operating system. This framework implements the networking functionality of the operating system.[6] The best known frameworks are the TCP/IP model and the OSI model.

At the time the Internet was developed, layering had proven to be a successful design approach for both compiler and operating system design and, given the similarities between programming languages and communication protocols, layering was applied to the protocols as well.[7] This gave rise to the concept of layered protocols which nowadays forms the basis of protocol design.[8]

Systems typically do not use a single protocol to handle a transmission. Instead they use a set of cooperating protocols, sometimes called a protocol family or protocol suite.[9] Some of the best known protocol suites include: IPX/SPX, X.25, AX.25, AppleTalk and TCP/IP.

The protocols can be arranged based on functionality in groups, for instance there is a group of transport protocols. The functionalities are mapped onto the layers, each layer solving a distinct class of problems relating to, for instance: application-, transport-, internet- and network interface-functions.[10] To transmit a message, a protocol has to be selected from each layer, so some sort of multiplexing/demultiplexing takes place. The selection of the next protocol is accomplished by extending the message with a protocolselector for each layer.[11]

Basic requirements of protocols

Messages are sent and received on communicating systems to establish communications. Protocols should therefore specify rules governing the transmission. In general, much of the following should be addressed:[12]

- *Data formats for data exchange*. Digital message bitstrings are exchanged. The bitstrings are divided in fields and each field carries information relevant to the protocol. Conceptually the bitstring is divided into two parts called the *header area* and the *data area*. The actual message is stored in the data area, so the header area contains the fields with more relevance to the protocol. Bitstrings longer than the maximum transmission unit (MTU) are divided in pieces of appropriate size.[13]
- *Address formats for data exchange*. Addresses are used to identify both the sender and the intended receiver(s). The addresses are stored in the header area of the bitstrings, allowing the receivers to determine whether the bitstrings are intended for themselves and should be processed or should be ignored. A connection between a sender and a receiver can be identified using an address pair *(sender address, receiver address)*. Usually some address values have special meanings. An all-*1*s address could be taken to mean an addressing of all stations on the network, so sending to this address would result in a broadcast on the local network. The rules describing the meanings of the address value are collectively called an *addressing scheme*.[14]
- *Address mapping*. Sometimes protocols need to map addresses of one scheme on addresses of another scheme. For instance to translate a logical IP address specified by the application to an Ethernet hardware address. This is

referred to as *address mapping*.[15]

- *Routing*. When systems are not directly connected, intermediary systems along the *route* to the intended receiver(s) need to forward messages on behalf of the sender. On the Internet, the networks are connected using routers. This way of connecting networks is called *internetworking*.
- *Detection of transmission errors* is necessary on networks which cannot guarantee error-free operation. In a common approach, CRCs of the data area are added to the end of packets, making it possible for the receiver to detect differences caused by errors. The receiver rejects the packets on CRC differences and arranges somehow for retransmission.[16]
- *Acknowledgements* of correct reception of packets is required for connection oriented communication. Acknowledgements are sent from receivers back to their respective senders.[17]
- *Loss of information - timeouts and retries*. Packets may be lost on the network or suffer from long delays. To cope with this, under some protocols, a sender may expect an acknowledgement of correct reception from the receiver within a certain amount of time. On timeouts, the sender must assume the packet was not received and retransmit it. In case of a permanently broken link, the retransmission has no effect so the number of retransmissions is limited. Exceeding the retry limit is considered an error.[18]
- *Direction of information flow* needs to be addressed if transmissions can only occur in one direction at a time as on half-duplex links. This is known as Media Access Control. Arrangements have to be made to accommodate the case when two parties want to gain control at the same time.[19]
- *Sequence control*. We have seen that long bitstrings are divided in pieces, and then sent on the network individually. The pieces may get lost or delayed or take different routes to their destination on some types of networks. As a result pieces may arrive out of sequence. Retransmissions can result duplicate pieces. By marking the pieces with sequence information at the sender, the receiver can determine what was lost or duplicated, ask for necessary retransmissions and reassemble the original message.[20]
- *Flow control* is needed when the sender transmits faster than the receiver or intermediate network equipment can process the transmissions. Flow control can be implemented by messaging from receiver to sender.[21]

Getting the data across a network is only part of the problem for a protocol. The data received has to be evaluated in the context of the progress of the conversation, so a protocol has to specify rules describing the context. These kind of rules are said to express the *syntax* of the communications. Other rules determine whether the data is meaningful for the context in which the exchange takes place. These kind of rules are said to express the *semantics* of the communications. Both intuitive descriptions as well as more formal specifications in the form of finite state machine models are used to describe the expected interactions of the protocol.[22] Formal ways for describing the syntax of the communications are Abstract Syntax Notation One (a ISO standard) or Augmented Backus-Naur form (a IETF standard).

Protocols and programming languages

> Protocols are to communications what algorithms or programming languages are to computations.[23] [1]

This analogy has important consequences for both the design and the development of protocols. One has to consider the fact that algorithms, programs and protocols are just different ways of describing expected behaviour of interacting objects. A familiar example of a protocolling language is the HTML language used to describe web pages which are the actual web protocols.

In programming languages the association of *identifiers* to a *value* is termed a *definition*. Program text is structured using *block* constructs and definitions can be local to a block. The localized association of an identifier to a value established by a definition is termed a *binding* and the region of program text in which a binding is effective is known as its *scope*.[24] The computational state is kept using two components: the *environment*, used as a record of identifier bindings, and the *store*, which is used as a record of the effects of assignments.[25]

In communications, message values are transferred using transmission media. By analogy, the equivalent of a store would be a collection of transmission media, instead of a collection of memory locations. A valid assignment in a protocol (as an analog of programming language) could be *Ethernet:='message'* , meaning a message is to be broadcast on the local ethernet.
On a transmission medium there can be many receivers. For instance a mac-address identifies an ether network card on the transmission medium (the 'ether'). In our imaginary protocol, the assignment *ethernet[mac-address]:=message value* could therefore make sense.[26]

By extending the assignment statement of an existing programming language with the semantics described, a protocolling language could easily be imagined.

Operating systems provide reliable communication and synchronization facilities for communicating objects confined to the same system by means of *system libraries*. A programmer using a general purpose programming language (like C or ADA) can use the routines in the libraries to implement a protocol, instead of using a dedicated protocolling language.

Universal protocols

The nice thing about standards is that you have so many to choose from.
—Andrew S. Tanenbaum in *Computer Networks*[27]

Despite their numbers, networking protocols show little variety, because all networking protocols use the same underlying principles and concepts, in the same way. So, the use of a general purpose programming language would yield a large number of applications only differing in the details.[28] A suitably defined (dedicated) protocolling language would therefore have little syntax, perhaps just enough to specify some parameters or optional modes of operation, because its virtual machine would have incorporated all possible principles and concepts making the virtual machine itself a *universal* protocol. The protocolling language would have some syntax and a lot of semantics describing this universal protocol and would therefore in effect be a protocol, hardly differing from this universal networking protocol. In this (networking) context a protocol is a language.

The notion of a universal networking protocol provides a rationale for standardization of networking protocols; assuming the existence of a universal networking protocol, development of protocol standards using a consensus model (the agreement of a group of experts) might be a viable way to coordinate protocol design efforts.

Networking protocols operate in very heterogeneous environments consisting of very different network technologies and a (possibly) very rich set of applications, so a single universal protocol would be very hard to design and implement correctly. Instead, the IETF decided to reduce complexity by assuming a relatively simple network architecture allowing decomposition of the single universal networking protocol into two generic protocols, TCP and IP, and two classes of specific protocols, one dealing with the low-level network details and one dealing with the high-level details of common network applications (remote login, file transfer, email and web browsing). ISO choose a similar but more general path, allowing other network architectures, to standardize protocols.

Protocol design

Communicating systems operate in parallel. The programming tools and techniques for dealing with parallel processes are collectively called *concurrent programming*. Concurrent programming only deals with the synchronization of communication. The syntax and semantics of the communication governed by a low-level protocol usually have modest complexity, so they can be coded with relative ease. High-level protocols with relatively large complexity could however merit the implementation of language interpreters. An example of the latter case is the HTML language.
Concurrent programming has traditionally been a topic in operating systems theorie texts.[29] Formal verification

seems indispensable, because concurrent programs are notorious for the hidden and sophisticated bugs they contain.[30] A mathematical approach to the study of concurrency and communication is referred to as *Communicating Sequential Processes* (CSP).[31] Concurrency can also be modelled using finite state machines like Mealy- and Moore machines. Mealy- and Moore machines are in use as design tools in digital electronics systems, which we encounter in the form of hardware used in telecommunications or electronic devices in general.[32]

This kind of design can be a bit of a challenge to say the least, so it is important to keep things simple. For the Internet protocols, in particular and in retrospect, this meant a basis for protocol design was needed to allow decomposition of protocols into much simpler, cooperating protocols.

Concurrent programming

A *concurrent program* is an abstraction of cooperating processes suitable for formal treatment and study. The goal of the abstraction is to prove correctness of the program assuming the existence of some basic synchronization or data exchange mechanisms provided by the operating system (or other software) or hardware. The mechanisms are complex, so more convenient higher level *primitives* are implemented with these mechanisms. The primitives are used to construct the concurrent program. The basic primitive for synchronization is the *semaphore*. All other primitives (locks, reentrant mutexes, semaphores, monitors, message passing, tuple space) can be defined using semaphores. The semaphore is sufficiently elementary to be successfully studied by formal methods.[33]

In order to synchronize or exchange data the processes must communicate by means of either a *shared memory*, used to store data or access-restricted procedures, or the sending/receiving of signals (*message passing*) using a shared transmission medium. Most third generation operating systems implement separate processes that use special instructions to ensure only one process can execute the restricted procedures. On distributed systems there is no common central memory so the communications are always by means of message passing. In this case the processes simply have to wait for each other (*synchronization by rendezvous*) before exchanging data.[4]

Conceptually, the concurrent program consists of several sequential processes whose execution sequences are interleaved. The execution sequences are divided into sections. A section manipulating shared resources is called a *critical section*. The interleaving scheme makes no timing assumptions other than that no process halts in its critical section and that ready processes are eventually scheduled for execution. For correct operation of the program, the critical sections of the processes need to be properly sequenced and synchronized. This is achieved using small code fragments (*protocols*) at the start and the end of the critical sections. The code fragments determine whether the critical sections of two communicating processes should execute in parallel (*rendezvous of processes*) or should be executed sequentially (*mutual exclusion of processes*).
A concurrent program is correct if it does not violate some safety property such as mutual exclusion or rendezvous of critical sections and does not suffer of liveness properties such as deadlock or lockout. Correctness of the concurrent program can only be shown using a mathematical argument. Specifications of concurrent programs can be formulated using formal logics (like CSP) which make it possible to prove properties of the programs. Incorrectness can be shown using execution scenarios.[5]

Mutual exclusion is extensively studied in the *mutual exclusion problem*. The rendezvous is studied in the *producer-consumer problem* in which a producer process only produces data if and only if the consumer process is ready to consume the data. Although both problems only involve two processes, their solutions require rather complex algorithms (Dekker's algorithm, Lamport's bakery algorithm). The readers-writers problem is a generalization of the mutual exclusion problem. The dining philosophers problem is a classical problem sufficiently difficult to expose many of the potential pitfalls of newly proposed primitives.[34]

A *basis* for protocol design

Systems do not use a single protocol to handle a transmission. Instead they use a set of cooperating protocols, sometimes called a protocol family or protocol suite.[9] To cooperate the protocols have to communicate with each other, so some kind of conceptual framework is needed to make this communication possible. Also note that software is needed to implement both the 'xfer-mechanism' and a protocol (no protocol, no communication).

In literature there are numerous references to the analogies between computer communication and programming. By analogy we could say that the aforementioned 'xfer-mechanism' is comparable to a *cpu*; a 'xfer-mechanism' performs communications and a *cpu* performs computations and the 'framework' introduces something that allows the protocols to be designed independent of one and another by providing separate execution environments for the protocols. Furthermore, it is repeatedly stated that *protocols are to computer communication what programming languages are to computation*.[35] [36]

Layering

Many communications protocols can be viewed as layered protocols where each layer gives a service to the protocol of the layers above and requires service from layers below.

Typically, a (best-effort) hardware delivery mechanism layer is used to build a connectionless packet delivery system on top of which a reliable transport layer is built, on top of which is the application software. Layers below and above these can be defined, and protocols are very often stacked to give *tunnelling*, for example the internet protocol can be tunnelled across an ATM network protocol to provide connectivity by layering the internet protocol on top of the ATM protocol transport layer.

The delivery system is defined by the IP protocol and the transport system by the TCP protocol.[37]

Protocol layering

Protocol layering now forms the basis of protocol design.[8] It allows the decomposition of single, complex protocols into simpler, cooperating protocols, but it is also a functional decomposition, because each protocol belongs to a functional class, called a *protocol layer*.[2] The protocol layers each solve a distinct class of communications problems. The Internet protocol suite consists of the following layers: application-, transport-, internet- and network interface-functions.[10] Together, the layers make up a *layering scheme* or *model*.

In computations, we have algorithms and data, and in communications, we have protocols and messages, so the analog of a data flow diagram would be some kind of message flow diagram.[23] To visualize protocol layering and protocol suites, a diagram of the message flows in and between two systems, A and B, is shown in figure 3.

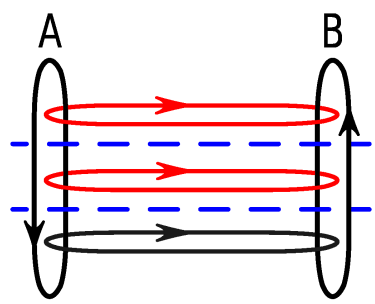


Figure 3. Message flows using a protocol suite.

The systems both make use of the same protocol suite. The vertical flows (and protocols) are *in system* and the horizontal message flows (and protocols) are *between* systems. The message flows are governed by rules, and dataformats specified by protocols. The blue lines therefore mark the boundaries of the (horizontal) protocol layers.

The vertical protocols are not layered because they don't obey the *protocol layering principle* which states that *a layered protocol is designed so that layer* n *at the destination receives exactly the same object sent by layer* n *at the source*. The horizontal protocols are *layered protocols* and all belong to the protocol suite. Layered protocols allow the protocol designer to concentrate on one layer at a time, without worrying about how other layers perform.[36]

The vertical protocols neednot be the same protocols on both systems, but they have to satisfy some minimal

assumptions to ensure the protocol layering principle holds for the layered protocols. This can be achieved using a technique called *Encapsulation*.[38]

Usually, a message or a stream of data is divided into small pieces, called *messages* or *streams*, *packets*, *IP datagrams* or *network frames* depending on the layer in which the pieces are to be transmitted. The pieces contain a *header area* and a *data area*. The data in the header area identifies the source and the destination on the network of the packet, the protocol, and other data meaningful to the protocol like CRC's of the data to be send, data length, and a timestamp.[39] [40]

The rule enforced by the vertical protocols is that the pieces for transmission are to be *encapsulated* in the data area of all lower protocols on the sending side and the reverse is to happen on the receiving side. The result is that at the lowest level the piece looks like this: 'Header1,Header2,Header3,data' and in the layer directly above it: 'Header2,Header3,data' and in the top layer: 'Header3,data', both on the sending and receiving side. This rule therefore ensures that the protocol layering principle holds and effectively virtualizes all but the lowest transmission lines, so for this reason some message flows are coloured red in figure 3.

To ensure both sides use the same protocol, the pieces also carry data identifying the protocol in their header.

The design of the protocol layering and the network (or Internet) architecture are interrelated, so one cannot be designed without the other.[41] Some of the more important features in this respect of the Internet architecture and the network services it provides are described next.

- The Internet offers *universal interconnection*, which means that any pair of computers connected to the Internet is allowed to communicate. Each computer is identified by an *address* on the Internet. All the interconnected physical networks appear to the user as a single large network. This interconnection scheme is called an *internetwork* or *internet*.[42]
- Conceptually, an *Internet addresses* consists of a *netid* and a *hostid*. The netid identifies a network and the hostid identifies a host. The term host is misleading in that an individual computer can have multiple network interfaces each having its own Internet address. An Internet Address identifies a connection to the network, not an individual computer.[43] The netid is used by routers to decide where to send a packet.[44]
- *Network technology independence* is achieved using the low-level *address resolution protocol* (ARP) which is used to map Internet addresses to physical addresses. The mapping is called *address resolution*. This way physical addresses are only used by the protocols of the network interface layer.[45] The TCP/IP protocols can make use of almost any underlying communication technology.[46]

Physical networks are interconnected by routers. Routers forward packets between interconnected networks making it possible for hosts to reach hosts on other physical networks. The message flows between two communicating system A and B in the presence of a router R are illustrated in figure 4. Datagrams are passed from router to router until a router is reached that can deliver the datagram on a physically attached network (called *direct delivery*).[47] To decide whether a datagram is to be delivered directly or is to be send to a router closer to the destination, a table called the *IP routing table* is consulted. The table consists of pairs of networkids and the paths to be taken to reach known networks. The path can be an indication that the datagram should be delivered directly or it can be the address of a router known to be closer to the destination.[48] A special entry can specify that a default router is chosen when there are no known paths.[49]

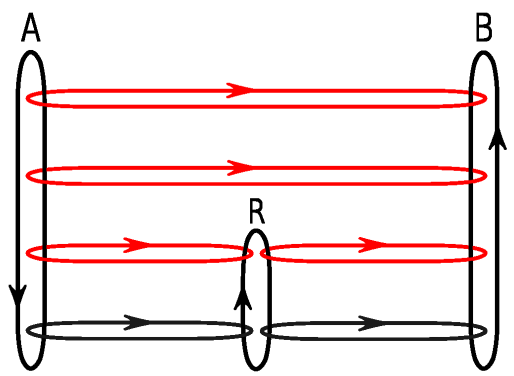


Figure 4. Message flows in the presence of a router

- *All networks are treated equal*. A LAN, a WAN or a point-to-point link between two computers are all considered as one network.[50]
- A *Connectionless packet delivery (or packet-switched) system (or service)* is offered by the Internet, because it adapts well to different hardware, including best-effort delivery mechanisms like the *ethernet*. Connectionless delivery means that the messages or streams are divided in pieces that are *multiplexed* separately on the high speed intermachine connections allowing the connections to be used concurrently. Each piece carries information identifying the destination. The delivery of packets is said to be *unreliable*, because packets may be lost, duplicated, delayed or delivered out of order without notice to the sender or receiver. Unreliability arises only when resources are exhausted or underlying networks fail.[51] The unreliable connectionless delivery system is defined by the *Internet Protocol* (IP). The protocol also specifies the *routing* function, which chooses a path over which data will be send.[52] It is also possible to use TCP/IP protocols on *connection oriented systems*. Connection oriented systems build up *virtual circuits* (paths for exclusive use) between senders and receivers. Once build up the IP datagrams are send as if they were data through the virtual circuits and forwarded (as data) to the IP protocol modules. This technique, called *tunneling*, can be used on X.25 networks and ATM networks.[53]
- A *reliable stream transport service* using the unreliable connectionless packet delivery service is defined by the *transmission control protocol* (TCP). The services are layered as well and the application programs residing in the layer above it, called the *application services*, can make use of TCP.[54] Programs wishing to interact with the packet delivery system itself can do so using the *user datagram protocol* (UDP).[55]

Software layering

Having established the protocol layering and the protocols, the protocol designer can now resume with the software design. The software has a layered organization and its relationship with protocol layering is visualized in figure 5.

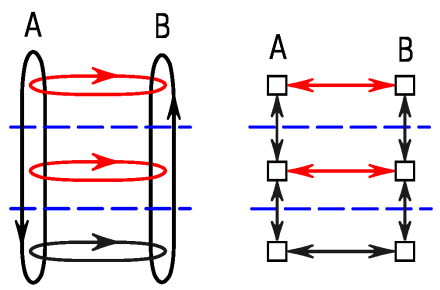


Figure 5: Protocol and software layering

The software modules implementing the protocols are represented by cubes. The information flow between the modules is represented by arrows. The (top two horizontal) red arrows are virtual. The blue lines mark the layer boundaries.

To send a message on system A, the top module interacts with the module directly below it and hands over the message to be encapsulated. This module reacts by encapsulating the message in its own data area and filling in its header data in accordance with the protocol it implements and interacts with the module below it by handing over this newly formed message whenever appropriate. The bottom module directly interacts with the bottom module of system B, so the message is send across. On the receiving system B the reverse happens, so ultimately (and assuming there were no transmission errors or protocol violations etc.) the message gets delivered in its original form to the topmodule of system B.[56]

On protocol errors, a receiving module discards the piece it has received and reports back the error condition to the original source of the piece on the same layer by handing the error message down or in case of the bottom module sending it across.[57]

The division of the message or stream of data into pieces and the subsequent reassembly are handled in the layer that introduced the division/reassembly. The reassembly is done at the destination (i.e. not on any intermediate routers).[58]

TCP/IP software is organized in four layers.[59]

- *Application layer*. At the highest layer, the services available across a TCP/IP internet are accessed by application programs. The application chooses the style of transport to be used which can be a sequence of individual messages or a continuous stream of bytes. The application program passes data to the *transport layer* for delivery.
- *Transport layer*. The transport layer provides communication from one application to another. The transport layer may regulate flow of information and provide reliable transport, ensuring that data arrives without error and in sequence. To do so, the receiving side sends back acknowledgments and the sending side retransmits lost pieces called packets. The stream of data is divided into packets by the module and each packet is passed along with a destination address to the next layer for transmission. The layer must accept data from many applications concurrently and therefore also includes codes in the packet header to identify the sending and receiving application program.
- *Internet layer*. The Internet layer handles the communication between machines. Packets to be send are accepted from the transport layer along with an identification of the receiving machine. The packets are encapsulated in IP datagrams and the datagram headers are filled. A routing algorithm is used to determine if the datagram should be delivered directly or send to a router. The datagram is passed to the appropriate network interface for transmission. Incoming datagrams are checked for validity and the routing algorithm is used to decide whether the datagram should be processed locally or forwarded. If the datagram is addressed to the local machine, the datagram header is deleted and the appropriate transport protocol for the packet is chosen. ICMP error and control messages are handled as well in this layer.
- *Network interface layer*. The network interface layer is responsible for accepting IP datagrams and transmitting them over a specific network. A network interface may consist of a device driver or a complex subsystem that uses its own data link protocol.

Program translation has been divided into four subproblems: compiler, assembler, link editor, and loader. As a result, the translation software is layered as well, allowing the software layers to be designed independently. Noting that the ways to conquer the complexity of program translation could readily be applied to protocols because of the analogy between programming languages and protocols. The designers of the TCP/IP protocol suite were keen on imposing the same layering on the software framework. This can be seen in the TCP/IP layering by considering the translation of a *pascal program* (message) that is compiled (function of the application layer) into an *assembler program* that is assembled (function of the transport layer) to *object code* (pieces) that is linked (function of the Internet layer) together with *library object code* (routing table) by the link editor, producing *relocatable machine code* (datagram) that is passed to the loader which fills in the memory locations (ethernet addresses) to produce *executeable code* (network frame) to be loaded (function of the network interface layer) into physical memory (transmission medium). To show just how closely the analogy fits, the terms between parentheses in the previous sentence denote the relevant analogs and the terms written *cursively* denote data representations. Program translation forms a linear sequence, because each layer's output is passed as input to the next layer. Furthermore, the translation process involves multiple data representations. We see the same thing happening in protocol software where multiple protocols define the datarepresentations of the data passed between the software modules.[3]

The network interface layer uses physical addresses and all the other layers only use IP addresses. The boundary between network interface layer and Internet layer is called the *high-level protocol address boundary*.[60] The modules below the application layer are generally considered part of the operating system. Passing data between these modules is much less expensive than passing data between an application program and the transport layer. The boundary between application layer and transport layer is called the *operating system boundary*.[61]

Strict layering

Strictly adhering to a layered model, a practice known as strict layering, is not always the best approach to networking.[62] Strict layering, can have a serious impact on the performance of the implementation, so there is at least a trade-off between simplicity and performance.[63] Another, perhaps more important point can be shown by considering the fact that some of the protocols in the Internet Protocol Suite cannot be expressed using the TCP/IP model, in other words some of the protocols behave in ways not described by the model.[64] To improve on the model, an offending protocol could, perhaps be split up into two protocols, at the cost of one or two extra layers, but there is a hidden caveat, because the model is also used to provide a conceptual view on the suite for the intended users. There is a trade-off to be made here between preciseness for the designer and clarity for the intended user.[65]

Protocol development

For communication to take place, protocols have to be agreed upon. Recall that in digital computing systems, the rules can be expressed by algorithms and datastructures, raising the opportunity of hardware independency. Expressing the algorithms in a portable programming language, makes the protocolsoftware operating system independent. The sourcecode could be considered a protocol specification. This form of specification, however is not suitable for the parties involved.

For one thing, this would enforce a source on all parties and for another, proprietary software producers would not accept this. By describing the software interfaces of the modules on paper and agreeing on the interfaces, implementers are free to do it their way. This is referred to as source independency. By specifying the algorithms on paper and detailing hardware dependencies in an unambiguous way, a *paper draft* is created, that when adhered to and published, ensures interoperability between software and hardware.

Such a paper draft can be developed into a *protocol standard* by getting the approval of a *standards organization*. To get the approval the paper draft needs to enter and successfully complete the *standardization process*. This activity is referred to as *protocol development*. The members of the standards organization agree to adhere to the standard on a voluntary basis. Often the members are in control of large market-shares relevant to the protocol and in many cases, standards are enforced by law or the government, because they are thought to serve an important public interest, so getting approval can be very important for the protocol. It should be noted though that in some cases protocol standards are not sufficient to gain widespread acceptance i.e. sometimes the sourcecode needs to be disclosed enforced by law or the government in the interest of the public.

The need for protocol standards

The need for protocol standards can be shown by looking at what happened to the bi-sync protocol (BSC) invented by IBM. BSC is an early link-level protocol used to connect two separate nodes. It was originally not intended to be used in a multinode network, but doing so revealed several deficiencies of the protocol. In the absence of standardization, manufacturers and organizations felt free to 'enhance' the protocol, creating incompatible versions on their networks. In some cases, this was deliberately done to discourage users from using equipment from other manufacturers. There are more than 50 variants of the original bi-sync protocol. One can assume, that a standard would have prevented at least some of this from happening.[6]

In some cases, protocols gain market dominance without going through a standardization process. Such protocols are referred to as *de facto standards*. De facto standards are common on emerging markets, niche markets, or markets that are monopolized (or oligopolized). They can hold a market in a very negative grip, especially when used to scare away competition. From a historical perspective, standardization should be seen as a measure to counteract the ill-effects of de facto standards. Positive exceptions exist; a 'de facto standard' operating system like GNU/Linux does not have this negative grip on its market, because the sources are published and maintained in an open way, thus inviting competition. Standardization is therefore not the only solution for *open systems interconnection*.

Standards organizations

Some of the standards organizations of relevance for communications protocols are the International Organization for Standardization (ISO), the International Telecommunications Union (ITU), the Institute of Electrical and Electronics Engineers (IEEE), and the Internet Engineering Task Force (IETF). The IETF maintains the protocols in use on the Internet. The IEEE controls many software and hardware protocols in the electronics industry for commercial and consumer devices. The ITU is an umbrella organization of telecommunications engineers designing the public switched telephone network (PSTN), as well as many radio communication systems. For marine electronics the NMEA standards are used. The World Wide Web Consortium (W3C) produces protocols and standards for Web technologies.

International standards organizations are supposed to be more impartial than local organizations with a national or commercial self-interest to consider. Standards organizations also do research and development for standards of the future. In practice, the standards organizations mentioned, cooperate closely with each other.[66]

The standardization process

The standardization process starts off with ISO commissioning a sub-committee workgroup. The workgroup issues working drafts and discussion documents to interested parties (including other standards bodies) in order to provoke discussion and comments. This will generate a lot of questions, much discussion and usually some disagreement on what the standard should provide and if it can satisfy all needs (usually not). All conflicting views should be taken into account, often by way of compromise, to progress to a *draft proposal* of the working group.
The draft proposal is discussed by the member countries' standard bodies and other organizations within each country. Comments and suggestions are collated and national views will be formulated, before the members of ISO vote on the proposal. If rejected, the draft proposal has to consider the objections and counter-proposals to create a new draft proposal for another vote. After a lot of feedback, modification, and compromise the proposal reaches the status of a *draft international standard*, and ultimately an *international standard*.
The process normally takes several years to complete. The original paper draft created by the designer will differ substantially from the standard, and will contain some of the following 'features':

- Various optional modes of operation, for example to allow for setup of different packet sizes at startup time, because the parties could not reach consensus on the optimum packet size.
- Parameters that are left undefined or allowed to take on values of a defined set at the discretion of the implementor. This often reflects conflicting views of some of the members.
- Parameters reserved for future use, reflecting that the members agreed the facility should be provided, but could not reach agreement on how this should be done in the available time.
- Various inconsistencies and ambiguities will inevitably be found when implementing the standard.

International standards are reissued periodically to handle the deficiencies and reflect changing views on the subject.[67]

Future of standardization (OSI)

A lesson learned from ARPANET (the predecessor of the Internet) is that standardization of protocols is not enough, because protocols also need a framework to operate. It is therefore important to develop a general-purpose, future-proof framework suitable for *structured protocols* (such as layered protocols) and their standardization. This would prevent protocol standards with overlapping functionality and would allow clear definition of the responsibilities of a protocol at the different levels (layers).[68] This gave rise to the ISO *Open Systems Interconnection reference model* (RM/OSI), which is used as a framework for the design of standard protocols and services conforming to the various layer specifications.[69]

In the OSI model, communicating systems are assumed to be connected by an underlying physical medium providing a basic (and unspecified) transmission mechanism. The layers above it are numbered (from one to seven);

the nth layer is referred to as (n)-layer. Each layer provides service to the layer above it (or at the top to the application process) using the services of the layer immediately below it. The layers communicate with each other by means of an interface, called a *service access point*. Corresponding layers at each system are called *peer entities*. To communicate, two peer entities at a given layer use a (n)-protocol, which is implemented by using services of the (n-1)-layer. When systems are not directly connected, intermediate peer entities (called *relays*) are used. An *address* uniquely identifies a service access point. The address naming domains need not be restricted to one layer, so it is possible to use just one naming domain for all layers.[70] For each layer there are two types of standards: protocol standards defining how peer entities at a given layer communicate, and service standards defining how a given layer communicates with the layer above it.

In the original version of RM/OSI, the layers and their functionality are (from highest to lowest layer):

- The *application layer* may provide the following services to the application processes: identification of the intended communication partners, establishment of the necessary authority to communicate, determination of availability and authentication of the partners, agreement on privacy mechanisms for the communication, agreement on responsibility for error recovery and procedures for ensuring data integrity, synchronization between cooperating application processes, identification of any constraints on syntax (e.g. character sets and data structures), determination of cost and acceptable quality of service, selection of the dialogue discipline, including required logon and logoff procedures.[71]
- The *presentation layer* may provide the following services to the application layer: a request for the establishment of a session, data transfer, negotiation of the syntax to be used between the application layers, any necessary syntax transformations, formatting and special purpose transformations (e.g. data compression and data encryption).[72]
- The *session layer* may provide the following services to the presentation layer: establishment and release of session connections, normal and expedited data exchange, a quarantine service which allows the sending presentation entity to instruct the receiving session entity not to release data to its presentation entity without permission, interaction management so presentation entities can control whose turn it is to perform certain control functions, resynchronization of a session connection, reporting of unrecoverable exceptions to the presentation entity.[73]
- The *transport layer* provides reliable and transparent data transfer in a cost effective way as required by the selected quality of service. It may support the multiplexing of several transport connections on to one network connection or split one transport connection into several network connections.[74]
- The *network layer* does the setup, maintenance and release of network paths between transport peer entities. When relays are needed, routing and relay functions are provided by this layer. The quality of service is negotiated between network and transport entities at the time the connection is set up. This layer is also responsible for (network) congestion control.[75]
- The *data link layer* does the setup, maintenance and release of data link connections. Errors occurring in the physical layer are detected and may be corrected. Errors are reported to the network layer. The exchange of data link units (including flow control) is defined by this layer.[76]
- The *physical layer* describes details like the electrical characteristics of the physical connection, the transmission techniques used, and the setup, maintenance and clearing of physical connections.[77]

In contrast to the TCP/IP layering scheme, which assumes a connectionless network, RM/OSI assumed a connection-oriented network. Connection-oriented networks are more suitable for wide area networks and connectionless networks are more suitable for local area networks. Using connections to communicate implies some form of session and (virtual) circuits, hence the (in the TCP/IP model lacking) session layer. The constituent members of ISO were mostly concerned with wide area networks, so development of RM/OSI concentrated on connection oriented networks and connectionless networks were only mentioned in an addendum to RM/OSI.[78] At the time, the IETF had to cope with this and the fact that the Internet needed protocols which simple were not there. As a result the IETF developed its own standardization process based on "rough consensus and running code".[79]

The standardization process is described by RFC2026 [80].
Nowadays, the IETF has become a standards organization for the protocols in use on the Internet. RM/OSI has extended its model to include connectionless services and because of this, both TCP and IP could be developed into international standards.

Taxonomies

Classification schemes for protocols usually focus on domain of use and function. As an example of domain of use, connection-oriented protocols and connectionless protocols are used on connection-oriented networks and connectionless networks respectively. For an example of function consider a tunneling protocol, which is used to encapsulate packets in a high-level protocol, so the packets can be passed across a transport system using the high-level protocol.

A *layering scheme* combines both function and domain of use. The dominant layering schemes are the ones proposed by the IETF and by ISO. Despite the fact that the underlying assumptions of the layering schemes are different enough to warrant distinguishing the two, it is a common practice to compare the two by relating common protocols to the layers of the two schemes.[81] For an example of this practice see: List of network protocols.

The layering scheme from the IETF is called *Internet layering* or *TCP/IP layering*. The functionality of the layers has been described in the section on software layering and an overview of protocols using this scheme is given in the article on Internet protocols.

The layering scheme from ISO is called *the OSI model* or *ISO layering*. The functionality of the layers has been described in the section on the future of standardization and an overview of protocols using this scheme is given in the article on OSI protocols.

Common types of protocols

The Internet Protocol is used in concert with other protocols within the Internet Protocol Suite. Prominent members of which include:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Control Message Protocol (ICMP)
- Hypertext Transfer Protocol (HTTP)
- Post Office Protocol (POP3)
- File Transfer Protocol (FTP)
- Internet Message Access Protocol (IMAP)

Other instances of high level interaction protocols are:

- IIOP
- RMI
- DCOM
- DDE
- SOAP

Notes

[1] Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, "They (protocols) are to communication what programming languages are to computation"

[2] Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, introduces the decomposition in layers.

[3] Comer 2000, Sect. 11.2 - The need for multiple protocols, p. 178, explains similarities protocol software and compiler, assembler, linker, loader.

[4] Ben-Ari 1982, chapter 2 - The concurrent programming abstraction, p. 18-19, states the same.

[5] Ben-Ari 1982, Section 2.7 - Summary, p. 27, summarizes the concurrent programming abstraction.

[6] Marsden 1986, Section 6.1 - Why are standards necessary?, p. 64-65, uses BSC as an example to show the need for both standard protocols and a standard framework.

[7] Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, explains this by drawing analogies between computer communication and programming languages.

[8] Sect. 11.10 - The Disadvantage Of Layering, p. 192, states: layering forms the basis for protocol design.

[9] Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, states the same.

[10] Comer 2000, Sect. 11.3 - The Conceptual Layers Of Protocol Software, p. 178, "Each layer takes responsibility for handling one part of the problem."

[11] Comer 2000, Sect. 11.11 - The Basic Idea Behind Multiplexing And Demultiplexing, p. 192, states the same.

[12] Marsden 1986, Chapter 3 - Fundamental protocol concepts and problem areas, p. 26-42, explains much of the following.

[13] Comer 2000, Sect. 7.7.4 - Datagram Size, Network MTU, and Fragmentation, p. 104, Explains fragmentation and the effect on the header of the fragments.

[14] Comer 2000, Chapter 4 - Classful Internet Addresses, p. 64-67;71.

[15] Marsden 1986, Section 14.3 - Layering concepts and general definitions, p. 187, explains address mapping.

[16] Marsden 1986, Section 3.2 - Detection and transmission errors, p. 27, explains the advantages of backward error correction.

[17] Marsden 1986, Section 3.3 - Acknowledgement, p. 28-33, explains the advantages of positive only acknowledgement and mentions datagram protocols as exceptions.

[18] Marsden 1986, Section 3.4 - Loss of information - timeouts and retries, p. 33-34.

[19] Marsden 1986, Section 3.5 - Direction of information flow, p. 34-35, explains master/slave and the negotiations to gain control.

[20] Marsden 1986, Section 3.6 - Sequence control, p. 35-36, explains how packets get lost and how sequencing solves this.

[21] Marsden 1986, Section 3.7 - Flow control, p. 36-38.

[22] Comer 2000, Glossary of Internetworking Terms and Abbreviations, p. 704, term protocol.

[23] Comer 2000, Sect. 1.3 - Internet Services, p. 3, "Protocols are to communication what algorithms are to computation"

[24] Tennent 1981, Section 2.3.1 - Definitions, p.15, defines scope and binding.

[25] Tennent 1981, Section 2.3.2 Environments and stores, p.16, the semantics of blocks and definitions are described using environments and stores.

[26] Hoare (1985), Ch. 4 - Communication, p. 133, In the introduction: a communication is an event described by a pair c.v where c is the name of the communication channel and v is the value of the message.

[27] Tanenbaum, Andrew S. (2003). *Computer networks* (http://books.google.com/books?id=Pd-z64SJRBAC&pg=PA235). Prentice Hall Professional. p. 235. ISBN 9780130661029. . Retrieved 22 June 2011.

[28] Comer 2000,Foreword To The First Edition By The Late Jon Postel, xxv, "The principles of architecture, layering, multiplexing, encapsulation, addressing and address mapping, routing, and naming are quite similar in any protocol suite, though of course, different in detail.".

[29] Ben-Ari 1982, in his preface, p. xiii.

[30] Ben-Ari 1982, in his preface, p. xiv.

[31] Hoare 1985, Chapter 4 - Communication, p. 133, deals with communication.

[32] S. Srinivasan, NPTEL courses:::: Electronics & Communication Engineering :: Digital Circuits and Systems, available online: http://nptel.iitm.ac.in/video.php?courseId=1005&p=3

[33] Ben-Ari 1982, Section 1.7 - An overview of the book, p. 12, states the same.

[34] Ben-Ari 1982, Section 4.3 - The producer-consumer problem, p. 56, explains the rendezvous variant in the absence of common memory.

[35] Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, states more or less the same, using other analogies.

[36] Comer 2000, Sect. 11.7 - The Protocol Layering Principle, p. 187, explains layered protocols.

[37] Comer 2000, Section 7.3 - Internet Architecture And Philosophy, p. 96, describes service layering.

[38] Comer 2000, Glossary of Internetworking terms, p.686: term encapsulation.

[39] Comer 2000, Sect. 11.5.1 - The TCP/IP 5-Layer Reference Model, p. 184, Describes the transformations of messages or streams that can be observed in the protocol layers.

[40] Comer 2000, Sect. 2.4.10 - Ethernet Frame Format, p. 30, Ethernet frames are used as an example for administrative data for the protocol itself.

[41] Comer 2000, Sect. 11.4 - Functionality Of The Layers, p. 181, states the same about the software organization.

[42] Comer 2000, Sect. 3.3 - Network-Level Interconnection, p. 55, explains universal interconnection and internetworking.

[43] Comer 2000, Sect. 4.4 - Addresses Specify Network Connections, p. 86, explains this.
[44] Comer 2000, Sect. 4.3 - The Original Classful Addressing Scheme, p. 64, explains the address scheme, netid and routing.
[45] Comer 2000, Sect. 5.13 - Summary, p. 86, explains ARP.
[46] Comer 2000, Sect. 2.11 - Other Technologies Over Which TCP/IP Has Been Used, p. 46, states the same.
[47] Comer 2000, Sect. 8.3.2 - Indirect Delivery, p. 118, states the same.
[48] Comer 2000, Sect. 8.5 - Next-Hop Routing, p. 120, gives details on the routing table.
[49] Comer 2000, Sect. 8.6 - Default Routes, p. 121, explains default routing and its use.
[50] Comer 2000, Sect. 3.8 - All Networks Are Equal, p. 59, states the same.
[51] Comer 2000, Sect. 7.5 - Connectionless Delivery System, p. 97, explains the delivery system.
[52] Comer 2000, Sect. 7.6 - Purposes Of The Internet Protocol, p. 97, states the same.
[53] Comer 2000, Sect. 2.11.1 - X25NET And Tunnels, p. 46-47, explains tunneling X.25 and mentions ATM.
[54] Comer 2000, Sect. 13.1 - Introduction, p. 209, introduces TCP.
[55] Comer 2000, Sect. 12.10 - Summary, p. 206, explains UDP.
[56] Comer 2000, Sect. 11.3 - The Conceptual Layers Of Protocol Software, p. 179, the first two paragraphs describe the sending of a message through successive layers.
[57] Comer 2000, Sect. 9.3 - Error Reporting vs. Error Correction, p. 131, describes the ICMP protocol that is used to handle datagram errors.
[58] Comer 2000, Sect. 7.7.5 - Reassembly Of Fragments, p. 104, describes reassembly of datagrams.
[59] Comer 2000, Sect. 11.5.1 - The TCP/IP 5-Layer Reference Model, p. 184, explains functionality of the layers.
[60] Comer 2000, Sect. 11.9.1 - High-Level Protocol Boundary, p. 191, describes the boundary.
[61] Comer 2000, Sect. 11.9.1 - Operating System Boundary, p. 192, describes the operating system boundary.
[62] IETF 1989, Sect 1.3.1 - Organization, p. 15, 2nd paragraph: many design choices involve creative "breaking" of strict layering.
[63] Comer 2000, Sect. 11.10 - The Disadvantage Of Layering, p. 192, explains why "strict layering can be extremely inefficient" giving examples of optimizations.
[64] IETF 1989, Sect 1.3.1 - Organization, p. 15, 2nd paragraph, explaining why "strict layering is an imperfect model"
[65] IETF 1989, Sect 1.3.1 - Organization, p. 15, states: This layerist organization was chosen for simplicity and clarity.
[66] Marsden 1986, Section 6.3 - Advantages of standardisation, p. 66-67, states the same.
[67] Marsden 1986, Section 6.4 - Some problems with standardisation, p. 67, follows HDLC to illustrate the process.
[68] Marsden 1986, Section 6.1 - Why are standards necessary?, p. 65, explains lessons learned from ARPANET.
[69] Marsden 1986, Section 14.1 - Introduction, p. 181, introduces OSI.
[70] Marsden 1986, Section 14.3 - Layering concepts and general definitions, p. 183-185, explains terminology.
[71] Marsden 1986, Section 14.4 - The application layer, p. 188, explains this.
[72] Marsden 1986, Section 14.5 - The presentation layer, p. 189, explains this.
[73] Marsden 1986, Section 14.6 - The session layer, p. 190, explains this.
[74] Marsden 1986, Section 14.7 - The transport layer, p. 191, explains this.
[75] Marsden 1986, Section 14.8 - The network layer, p. 192, explains this.
[76] Marsden 1986, Section 14.9 - The data link layer, p. 194, explains this.
[77] Marsden 1986, Section 14.10 - The physical layer, p. 195, explains this.
[78] Marsden 1986, Section 14.11 - Connectionless mode and RM/OSI, p. 195, mentions this.
[79] Comer 2000, Section 1.9 - Internet Protocols And Standardization, p. 12, explains why the IETF did not use existing protocols.
[80] http://tools.ietf.org/html/rfc2026
[81] Comer 2000, Sect. 11.5.1 - The TCP/IP 5-Layer Reference Model, p. 183, states the same.

References

- Radia Perlman: *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols.* 2nd Edition. Addison-Wesley 1999, ISBN 0-201-63448-1. In particular Ch. 18 on "network design folklore", which is also available online at http://www.informit.com/articles/article.aspx?p=20482
- Gerard J. Holzmann: *Design and Validation of Computer Protocols.* Prentice Hall, 1991, ISBN 0-13-539925-4. Also available online at http://spinroot.com/spin/Doc/Book91.html
- Douglas E. Comer (2000). *Internetworking with TCP/IP - Principles, Protocols and Architecture* (4th ed.). Prentice Hall. ISBN 0-13-018380-6. In particular Ch.11 Protocol layering. Also has a RFC guide and a Glossary of Internetworking Terms and Abbreviations.
- Internet Engineering Task Force abbr. IETF (1989): *RFC1122, Requirements for Internet Hosts -- Communication Layers, R.Braden (ed.)*, Available online at http://tools.ietf.org/html/rfc1122.Describes TCP/IP to the implementors of protocolsoftware. In particular the introduction gives an overview of the design goals of the suite.

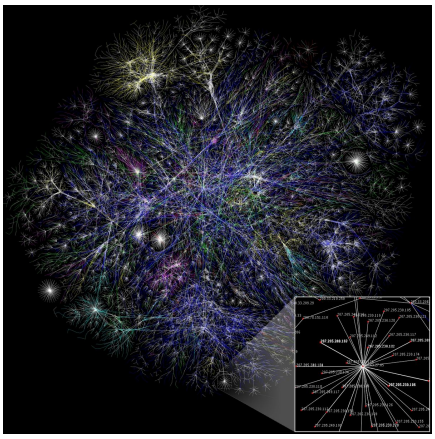
- M. Ben-Ari (1982): *Principles of concurrent programming* 10th Print. Prentice Hall International, ISBN 0-13-701078-8.
- C.A.R. Hoare (1985): *Communicating sequential processes* 10th Print. Prentice Hall International, ISBN 0-13-153271-5. Available online via http://www.usingcsp.com
- R.D. Tennent (1981): *Principles of programming languages* 10th Print. Prentice Hall International, ISBN 0-13-709873-1.
- Brian W Marsden (1986): *Communication network protocols* 2nd Edition. Chartwell Bratt, ISBN 0-86238-106-1.
- Andrew S. Tanenbaum (1984): *Structured computer organization* 10th Print. Prentice Hall International, ISBN 0-13-854605-3.

External links

- Javvin's Protocol Dictionary (http://www.javvin.com/protocolsuite.html)
- Network Protocol Index (http://www.networkdictionary.com/protocols/)
- Overview of protocols in telecontrol field with OSI Reference Model (http://www.ipcomm.de/protocols_en.html)
- List of Data Communication Protocols (http://www.zframez.com/protocolsuite.html)
- PDF-Chart showing the Protocols and the OSI reference layer (http://www.wildpackets.com/elements/misc/WP_encapsulation_chart.pdf)

Computer network

A **computer network**, often simply referred to as a network, is a collection of hardware components and computers interconnected by communication channels that allow sharing of resources and information.[1]



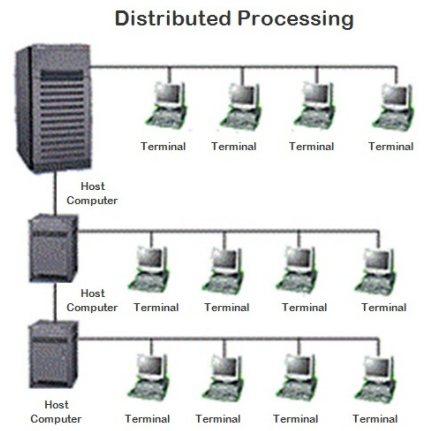
Internet map. The Internet is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide.

Technically, Network is defined as "If at least one process in one computer is able to send/receive data to/from at least one process residing in a remote computer, then the two computers are said to be in network".

Networks may be classified according to a wide variety of characteristics such as the medium used to transport the data, communications protocol used, scale, topology, and organizational scope.

The rules and data formats for exchanging information in a computer network are defined by communications protocols. Well-known communications protocols are Ethernet, a hardware and Link Layer standard that is ubiquitous in local area networks, and the Internet Protocol Suite, which defines a set of protocols for internetworking, i.e. for data communication between multiple networks, as well as host-to-host data transfer, and application-specific data transmission formats.

Computer networking is sometimes considered a sub-discipline of electrical engineering, telecommunications, computer science, information technology or computer engineering, since it relies upon the theoretical and practical application of these disciplines.



Distributed processing

History

Before the advent of computer networks that were based upon some type of telecommunications system, communication between calculation machines and early computers was performed by human users by carrying instructions between them. Many of the social behaviors seen in today's Internet were demonstrably present in the 19th century and arguably in even earlier networks using visual signals.

- In September 1940, George Stibitz used a Teletype machine to send instructions for a problem set from his Model at Dartmouth College to his Complex Number Calculator in New York and received results back by the same means. Linking output systems like teletypewriters to computers was an interest at the Advanced Research Projects Agency (ARPA) when, in 1962, J.C.R. Licklider was hired and developed a working group he called the "Intergalactic Network", a precursor to the ARPANET.
- Early networks of communicating computers included the military radar system Semi-Automatic Ground Environment (SAGE), started in the late 1950s
- The commercial airline reservation system semi-automatic business research environment (SABRE) which went online with two connected mainframes in 1960.[2] [3]
- In 1964, researchers at Dartmouth developed the Dartmouth Time Sharing System for distributed users of large computer systems. The same year, at Massachusetts Institute of Technology, a research group supported by General Electric and Bell Labs used a computer to route and manage telephone connections.
- Throughout the 1960s Leonard Kleinrock, Paul Baran and Donald Davies independently conceptualized and developed network systems which used packets that could be used in a network between computer systems.
- 1965 Thomas Merrill and Lawrence G. Roberts created the first wide area network (WAN).
- The first widely used telephone switch that used true computer control was introduced by Western Electric in 1965.
- In 1969 the University of California at Los Angeles, the Stanford Research Institute, University of California at Santa Barbara, and the University of Utah were connected as the beginning of the ARPANET network using 50 kbit/s circuits.[4]
- Commercial services using X.25 were deployed in 1972, and later used as an underlying infrastructure for expanding TCP/IP networks.

Today, computer networks are the core of modern communication. All modern aspects of the public switched telephone network (PSTN) are computer-controlled, and telephony increasingly runs over the Internet Protocol, although not necessarily the public Internet. The scope of communication has increased significantly in the past decade, and this boom in communications would not have been possible without the progressively advancing computer network. Computer networks, and the technologies needed to connect and communicate through and between them, continue to drive computer hardware, software, and peripherals industries. This expansion is mirrored by growth in the numbers and types of users of networks from the researcher to the home user.

Properties

Computer networks:

Facilitate communications

Using a network, people can communicate efficiently and easily via email, instant messaging, chat rooms, telephone, video telephone calls, and video conferencing.

Permit sharing of files, data, and other types of information

In a network environment, authorized users may access data and information stored on other computers on the network. The capability of providing access to data and information on shared storage devices is an important feature of many networks.

Share network and computing resources

In a networked environment, each computer on a network may access and use resources provided by devices on the network, such as printing a document on a shared network printer. Distributed computing uses computing resources across a network to accomplish tasks.

May be insecure

A computer network may be used by computer hackers to deploy computer viruses or computer worms on devices connected to the network, or to prevent these devices from normally accessing the network (denial of service).

May interfere with other technologies

Power line communication strongly disturbs certain forms of radio communication, e.g., amateur radio.[5] It may also interfere with last mile access technologies such as ADSL and VDSL.[6]

May be difficult to set up

A complex computer network may be difficult to set up. It may also be very costly to set up an effective computer network in a large organization or company.

Communication media

Computer networks can be classified according to the hardware and associated software technology that is used to interconnect the individual devices in the network, such as electrical cable (HomePNA, power line communication, G.hn), optical fiber, and radio waves (wireless LAN). In the OSI model, these are located at levels 1 and 2.

A well-known *family* of communication media is collectively known as Ethernet. It is defined by IEEE 802 and utilizes various standards and media that enable communication between devices. Wireless LAN technology is designed to connect devices without wiring. These devices use radio waves or infrared signals as a transmission medium.

Wired technologies

- *Twisted pair wire* is the most widely used medium for telecommunication. Twisted-pair cabling consist of copper wires that are twisted into pairs. Ordinary telephone wires consist of two insulated copper wires twisted into pairs. Computer networking cabling (wired Ethernet as defined by IEEE 802.3) consists of 4 pairs of copper cabling that can be utilized for both voice and data transmission. The use of two wires twisted together helps to reduce crosstalk and electromagnetic induction. The transmission speed ranges from 2 million bits per second to 10 billion bits per second. Twisted pair cabling comes in two forms which are Unshielded Twisted Pair (UTP) and Shielded twisted-pair (STP) which are rated in categories which are manufactured in different increments for various scenario.

- *Coaxial cable* is widely used for cable television systems, office buildings, and other work-sites for local area networks. The cables consist of copper or aluminum wire wrapped with insulating layer typically of a flexible material with a high dielectric constant, all of which are surrounded by a conductive layer. The layers of insulation help minimize interference and distortion. Transmission speed range from 200 million to more than 500 million bits per second.
- ITU-T G.hn technology uses existing home wiring (coaxial cable, phone lines and power lines) to create a high-speed (up to 1 Gigabit/s) local area network.
- *Optical fiber cable* consists of one or more filaments of glass fiber wrapped in protective layers that carries data by means of pulses of light. It transmits light which can travel over extended distances. Fiber-optic cables are not affected by electromagnetic radiation. Transmission speed may reach trillions of bits per second. The transmission speed of fiber optics is hundreds of times faster than for coaxial cables and thousands of times faster than a twisted-pair wire. This *capacity* may be further increased by the use of colored light, i.e., light of multiple wavelengths. Instead of carrying one message in a stream of monochromatic light impulses, this technology can carry multiple signals in a single fiber.

Wireless technologies

- *Terrestrial microwave* – Terrestrial microwaves use Earth-based transmitter and receiver. The equipment looks similar to satellite dishes. Terrestrial microwaves use low-gigahertz range, which limits all communications to line-of-sight. Path between relay stations spaced approx, 48 km (30 mi) apart.
- *Communications satellites* – The satellites use microwave radio as their telecommunications medium which are not deflected by the Earth's atmosphere. The satellites are stationed in space, typically **unknown operator: u','unknown operator: u','unknown operator: u','** (**unknown operator: u'strong'unknown operator: u','**mi) (for geosynchronous satellites) above the equator. These Earth-orbiting systems are capable of receiving and relaying voice, data, and TV signals.
- *Cellular and PCS systems* – Use several radio communications technologies. The systems are divided to different geographic areas. Each area has a low-power transmitter or radio relay antenna device to relay calls from one area to the next area.
- *Wireless LANs* – Wireless local area network use a high-frequency radio technology similar to digital cellular and a low-frequency radio technology. Wireless LANs use spread spectrum technology to enable communication between multiple devices in a limited area. An example of open-standards wireless radio-wave technology is IEEE 802.11.
- Infrared communication can transmit signals between devices within small distances of typically no more than 10 meters. In most cases, line-of-sight propagation is used, which limits the physical positioning of communicating devices.
- A global area network (GAN) is a network used for supporting mobile across an arbitrary number of wireless LANs, satellite coverage areas, etc. The key challenge in mobile communications is handing off the user communications from one local coverage area to the next. In IEEE Project 802, this involves a succession of terrestrial wireless LANs.[7]

Exotic technologies

There have been various attempts at transporting data over more or less exotic media:

- IP over Avian Carriers was a humorous April fool's Request for Comments, issued as **RFC 1149**. It was implemented in real life in 2001.[8]
- Extending the Internet to interplanetary dimensions via radio waves.[9]

A practical limit in both cases is the round-trip delay time which constrains useful communication.

Communications protocol

A communications protocol defines the formats and rules for exchanging information via a network and typically comprises a complete protocol suite which describes the protocols used at various usage levels. An interesting feature of communications protocols is that they may be – and in fact very often are – stacked above each other, which means that one is used to carry the other. *The* example for this is HTTP running over TCP over IP over IEEE 802.11, where the second and third are members of the Internet Protocol Suite, while the last is a member of the Ethernet protocol suite. This is the stacking which exists between the wireless router and the home user's personal computer when surfing the World Wide Web.

Communication protocols have themselves various properties, such as whether they are connection-oriented versus connectionless, whether they use circuit mode or packet switching, or whether they use hierarchical or flat addressing.

There exist a multitude of communication protocols, a few of which are described below.

Ethernet

Ethernet is a family of connectionless protocols used in LANs, described by a set of standards together called IEEE 802 published by the Institute of Electrical and Electronics Engineers. It has a flat addressing scheme and is mostly situated at levels 1 and 2 of the OSI model. For home users today, the most well-known member of this protocol family is IEEE 802.11, otherwise known as Wireless LAN (WLAN). However, the complete protocol suite deals with a multitude of networking aspects not only for home use, but especially when the technology is deployed to support a diverse range of business needs. MAC bridging (IEEE 802.1D) deals with the routing of Ethernet packets using a Spanning Tree Protocol, IEEE 802.1Q describes VLANs, and IEEE 802.1X defines a port-based Network Access Control protocol which forms the basis for the authentication mechanisms used in VLANs, but also found in WLANs – it is what the home user sees when they have to enter a "wireless access key".

Internet Protocol Suite

The Internet Protocol Suite, often also called TCP/IP, is the foundation of all modern internetworking. It offers connection-less as well as connection-oriented services over an inherently unreliable network traversed by datagram transmission at the Internet protocol (IP) level. At its core, the protocol suite defines the addressing, identification, and routing specification in form of the traditional Internet Protocol Version 4 (IPv4) and IPv6, the next generation of the protocol with a much enlarged addressing capability.

SONET/SDH

Synchronous Optical NETworking (SONET) and Synchronous Digital Hierarchy (SDH) are standardized multiplexing protocols that transfer multiple digital bit streams over optical fiber using lasers. They were originally designed to transport circuit mode communications from a variety of different sources, primarily to support real-time, uncompressed, circuit-switched voice encoded in PCM format. However, due to its protocol neutrality and transport-oriented features, SONET/SDH also was the obvious choice for transporting Asynchronous Transfer Mode (ATM) frames.

Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a switching technique for telecommunication networks. It uses asynchronous time-division multiplexing and encodes data into small, fixed-sized cells. This differs from other protocols such as the Internet Protocol Suite or Ethernet that use variable sized packets or frames. ATM has similarity with both circuit and packet switched networking. This makes it a good choice for a network that must handle both traditional high-throughput data traffic, and real-time, low-latency content such as voice and video. ATM uses a connection-oriented model in which a virtual circuit must be established between two endpoints before the actual data exchange begins.

While the role of ATM is diminishing in favor of next-generation networks, it still plays a role in the last mile, which is the connection between an Internet service provider and the home user. For an interesting write-up of the technologies involved, including the deep stacking of communications protocols used, see.[10]

Scale

Computer network types by geographical scope

• Near field (NFC)
• Body (BAN)
• Personal (PAN)
• Near-me (NAN)
• Local (LAN)
  • Home (HAN)
  • Storage (SAN)
• Campus (CAN)
• Backbone
• Metropolitan (MAN)
• Wide (WAN)
• Internet
• Interplanetary Internet

Networks are often classified by their physical or organizational extent or their purpose. Usage, trust level, and access rights differ between these types of networks.

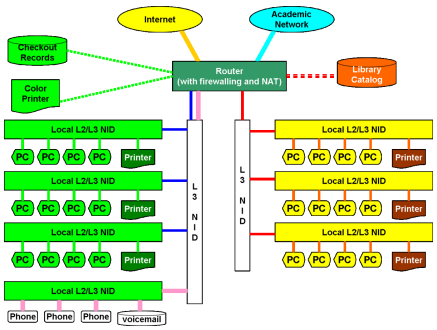
Personal area network

A personal area network (PAN) is a computer network used for communication among computer and different information technological devices close to one person. Some examples of devices that are used in a PAN are personal computers, printers, fax machines, telephones, PDAs, scanners, and even video game consoles. A PAN may include wired and wireless devices. The reach of a PAN typically extends to 10 meters.[11] A wired PAN is usually constructed with USB and Firewire connections while technologies such as Bluetooth and infrared communication typically form a wireless PAN.

Local area network

A local area network (LAN) is a network that connects computers and devices in a limited geographical area such as home, school, computer laboratory, office building, or closely positioned group of buildings. Each computer or device on the network is a node. Current wired LANs are most likely to be based on Ethernet technology, although new standards like ITU-T G.hn also provide a way to create a wired LAN using existing home wires (coaxial cables, phone lines and power lines).[12]

All interconnected devices must understand the network layer (layer 3), because they are handling multiple subnets (the different colors). Those inside the library, which have only 10/100 Mbit/s Ethernet connections to the user device and a Gigabit Ethernet connection to the central router, could be called "layer 3 switches" because they only have Ethernet interfaces and must understand IP. It would be more correct to call them access routers, where the router at the top is a distribution router that connects to the Internet and academic networks' customer access routers.



Typical library network, in a branching tree topology and controlled access to resources

The defining characteristics of LANs, in contrast to WANs (Wide Area Networks), include their higher data transfer rates, smaller geographic range, and no need for leased telecommunication lines. Current Ethernet or other IEEE 802.3 LAN technologies operate at speeds up to 10 Gbit/s. This is the data transfer rate. IEEE has projects investigating the standardization of 40 and 100 Gbit/s.[13] LANs can be connected to Wide area network by using routers.

Home network

A home network is a residential LAN which is used for communication between digital devices typically deployed in the home, usually a small number of personal computers and accessories, such as printers and mobile computing devices. An important function is the sharing of Internet access, often a broadband service through a cable TV or Digital Subscriber Line (DSL) provider.

Storage area network

A storage area network (SAN) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to make storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

Campus network

A campus network is a computer network made up of an interconnection of LANs within a limited geographical area. The networking equipment (switches, routers) and transmission media (optical fiber, copper plant, Cat5 cabling etc.) are almost entirely owned (by the campus tenant / owner: an enterprise, university, government etc.).

In the case of a university campus-based campus network, the network is likely to link a variety of campus buildings including, for example, academic colleges or departments, the university library, and student residence halls.

Backbone network

A backbone network is part of a computer network infrastructure that interconnects various pieces of network, providing a path for the exchange of information between different LANs or subnetworks. A backbone can tie together diverse networks in the same building, in different buildings in a campus environment, or over wide areas. Normally, the backbone's capacity is greater than that of the networks connected to it.

A large corporation which has many locations may have a backbone network that ties all of these locations together, for example, if a server cluster needs to be accessed by different departments of a company which are located at different geographical locations. The equipment which ties these departments together constitute the network backbone. Network performance management including network congestion are critical parameters taken into account when designing a network backbone.

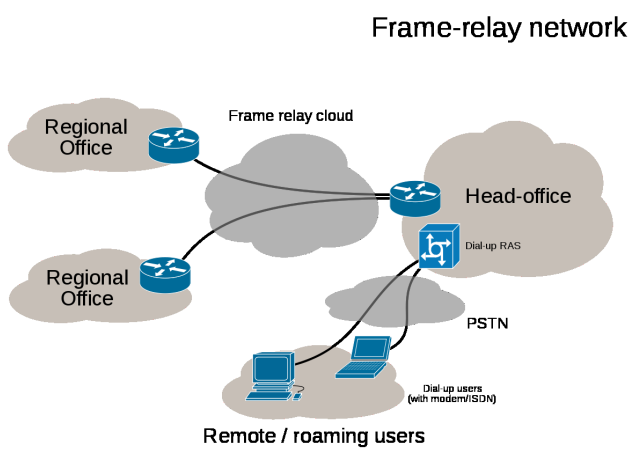
A specific case of a backbone network is the Internet backbone, which is the set of wide-area network connections and core routers that interconnect all networks connected to the Internet.

Metropolitan area network

A Metropolitan area network (MAN) is a large computer network that usually spans a city or a large campus.

Wide area network

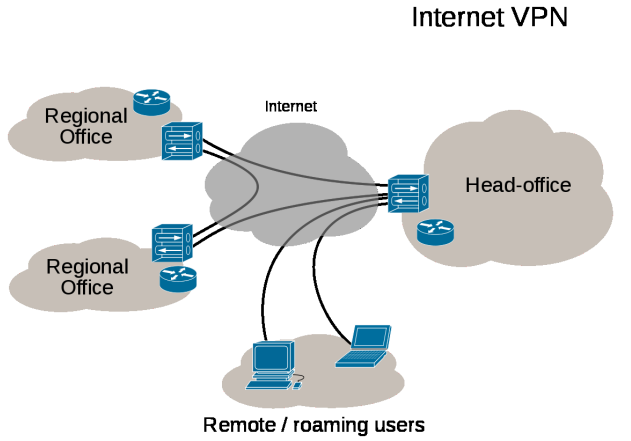
A wide area network (WAN) is a computer network that covers a large geographic area such as a city, country, or spans even intercontinental distances, using a communications channel that combines many types of media such as telephone lines, cables, and air waves. A WAN often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.



Sample EPN made of Frame relay WAN connections and dialup remote access.

Enterprise private network

An enterprise private network is a network built by an enterprise to interconnect various company sites, e.g., production sites, head offices, remote offices, shops, in order to share computer resources.



Sample VPN used to interconnect 3 offices and remote users

Virtual private network

A virtual private network (VPN) is a computer network in which some of the links between nodes are carried by open connections or virtual circuits in some larger network (e.g., the Internet) instead of by physical wires. The data link layer protocols

of the virtual network are said to be tunneled through the larger network when this is the case. One common application is secure communications through the public Internet, but a VPN need not have explicit security features, such as authentication or content encryption. VPNs, for example, can be used to separate the traffic of different user communities over an underlying network with strong security features.

VPN may have best-effort performance, or may have a defined service level agreement (SLA) between the VPN customer and the VPN service provider. Generally, a VPN has a topology more complex than point-to-point.

Internetwork

An internetwork is the connection of multiple computer networks via a common routing technology using routers. The Internet is an aggregation of many connected internetworks spanning the Earth.

Organizational scope

Networks are typically managed by organizations which own them. According to the owner's point of view, networks are seen as intranets or extranets. A special case of network is the Internet, which has no single owner but a distinct status when seen by an organizational entity – that of permitting virtually unlimited global connectivity for a great multitude of purposes.

Intranets and extranets

Intranets and extranets are parts or extensions of a computer network, usually a LAN.

An intranet is a set of networks, using the Internet Protocol and IP-based tools such as web browsers and file transfer applications, that is under the control of a single administrative entity. That administrative entity closes the intranet to all but specific, authorized users. Most commonly, an intranet is the internal network of an organization. A large intranet will typically have at least one web server to provide users with organizational information.

An extranet is a network that is limited in scope to a single organization or entity and also has limited connections to the networks of one or more other usually, but not necessarily, trusted organizations or entities—a company's customers may be given access to some part of its intranet—while at the same time the customers may not be considered *trusted* from a security standpoint. Technically, an extranet may also be categorized as a CAN, MAN, WAN, or other type of network, although an extranet cannot consist of a single LAN; it must have at least one connection with an external network.

Internet

The Internet is a global system of interconnected governmental, academic, corporate, public, and private computer networks. It is based on the networking technologies of the Internet Protocol Suite. It is the successor of the Advanced Research Projects Agency Network (ARPANET) developed by DARPA of the United States Department of Defense. The Internet is also the communications backbone underlying the World Wide Web (WWW).

Participants in the Internet use a diverse array of methods of several hundred documented, and often standardized, protocols compatible with the Internet Protocol Suite and an addressing system (IP addresses) administered by the Internet Assigned Numbers Authority and address registries. Service providers and large enterprises exchange information about the reachability of their address spaces through the Border Gateway Protocol (BGP), forming a redundant worldwide mesh of transmission paths.

Network topology

Common layouts

A network topology is the layout of the interconnections of the nodes of a computer network. Common layouts are:

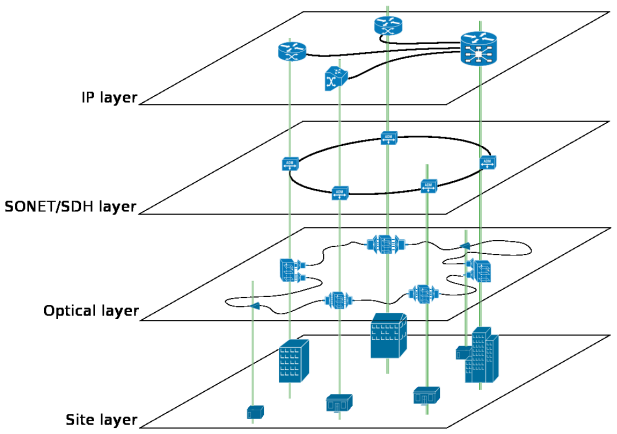
- A bus network: all nodes are connected to a common medium along this medium. This was the layout used in the original Ethernet, called 10BASE5 and 10BASE2.
- A star network: all nodes are connected to a special central node. This is the typical layout found in in a Wireless LAN, where each wireless client connects to the central Wireless access point.
- A ring network: each node is connected to its left and right neighbor node, such that all nodes are connected and that each node can reach each other node by traversing nodes left- or rightwards. The Fiber Distributed Data Interface (FDDI) made use of such a topology.
- A mesh network: each node is connected to an arbitrary number of neighbors in such a way that there is at least one traversal from any node to any other.
- A fully connected network: each node is connected to every other node in the network.

Note that the physical layout of the nodes in a network may not necessarily reflect the network topology. As an example, with FDDI, the network topology is a ring (actually two counter-rotating rings), but the physical topology is a star, because all neighboring connections are routed via a central physical location.

Overlay network

An overlay network is a virtual computer network that is built on top of another network. Nodes in the overlay are connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. The topology of the overlay network may (and often does) differ from that of the underlying one.

For example, many peer-to-peer networks are overlay networks because they are organized as nodes of a virtual system of links run on top of the Internet. The Internet was initially built as an overlay on the telephone network.[14]



A sample overlay network: IP over SONET over Optical

The most striking example of an overlay network, however, is the Internet itself: At the IP layer, each node can reach any other by a direct connection to the desired IP address, thereby creating a fully connected network; the underlying network, however, is composed of a mesh-like interconnect of subnetworks of varying topologies (and, in fact, technologies). Address resolution and routing are the means which allows the mapping of the fully connected IP overlay network to the underlying ones.

Overlay networks have been around since the invention of networking when computer systems were connected over telephone lines using modems, before any data network existed.

Another example of an overlay network is a distributed hash table, which maps keys to nodes in the network. In this case, the underlying network is an IP network, and the overlay network is a table (actually a map) indexed by keys.

Overlay networks have also been proposed as a way to improve Internet routing, such as through quality of service guarantees to achieve higher-quality streaming media. Previous proposals such as IntServ, DiffServ, and IP Multicast have not seen wide acceptance largely because they require modification of all routers in the network. On the other

hand, an overlay network can be incrementally deployed on end-hosts running the overlay protocol software, without cooperation from Internet service providers. The overlay has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes a message traverses before reaching its destination.

For example, Akamai Technologies manages an overlay network that provides reliable, efficient content delivery (a kind of multicast). Academic research includes end system multicast [15] and overcast for multicast; RON (resilient overlay network) for resilient routing; and OverQoS for quality of service guarantees, among others.

Basic hardware components

Apart from the physical communications media themselves as described above, networks comprise additional basic hardware building blocks interconnecting their terminals, such as network interface cards (NICs), hubs, bridges, switches, and routers.

Network interface cards

A network card, network adapter, or NIC (network interface card) is a piece of computer hardware designed to allow computers to physically access a networking medium. It provides a low-level addressing system through the use of MAC addresses.

Each Ethernet network interface has a unique MAC address which is usually stored in a small memory device on the card, allowing any device to connect to the network without creating an address conflict. Ethernet MAC addresses are composed of six octets. Uniqueness is maintained by the IEEE, which manages the Ethernet address space by assigning 3-octet prefixes to equipment manufacturers. The list of prefixes [16] is publicly available. Each manufacturer is then obliged to both use only their assigned prefix(es) and to uniquely set the 3-octet suffix of every Ethernet interface they produce.

Repeaters and hubs

A repeater is an electronic device that receives a signal, cleans it of unnecessary noise, regenerates it, and retransmits it at a higher power level, or to the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable that runs longer than 100 meters. A repeater with multiple ports is known as a hub. Repeaters work on the Physical Layer of the OSI model. Repeaters require a small amount of time to regenerate the signal. This can cause a propagation delay which can affect network communication when there are several repeaters in a row. Many network architectures limit the number of repeaters that can be used in a row (e.g. Ethernet's 5-4-3 rule).

Today, repeaters and hubs have been made mostly obsolete by switches (see below).

Bridges

A network bridge connects multiple network segments at the data link layer (layer 2) of the OSI model. Bridges broadcast to all ports except the port on which the broadcast was received. However, bridges do not promiscuously copy traffic to all ports, as hubs do, but learn which MAC addresses are reachable through specific ports. Once the bridge associates a port and an address, it will send traffic for that address to that port only.

Bridges learn the association of ports and addresses by examining the source address of frames that it sees on various ports. Once a frame arrives through a port, its source address is stored and the bridge assumes that MAC address is associated with that port. The first time that a previously unknown destination address is seen, the bridge will forward the frame to all ports other than the one on which the frame arrived.

Bridges come in three basic types:

- Local bridges: Directly connect LANs

- Remote bridges: Can be used to create a wide area network (WAN) link between LANs. Remote bridges, where the connecting link is slower than the end networks, largely have been replaced with routers.
- Wireless bridges: Can be used to join LANs or connect remote stations to LANs.

Switches

A network switch is a device that forwards and filters OSI layer 2 datagrams (chunks of data communication) between ports (connected cables) based on the MAC addresses in the packets.[17] A switch is distinct from a hub in that it only forwards the frames to the ports involved in the communication rather than all ports connected. A switch breaks the collision domain but represents itself as a broadcast domain. Switches make forwarding decisions of frames on the basis of MAC addresses. A switch normally has numerous ports, facilitating a star topology for devices, and cascading additional switches.[18] Some switches are capable of routing based on Layer 3 addressing or additional logical levels; these are called multi-layer switches. The term *switch* is used loosely in marketing to encompass devices including routers and bridges, as well as devices that may distribute traffic on load or by application content (e.g., a Web URL identifier).

Routers

A router is an internetworking device that forwards packets between networks by processing information found in the datagram or packet (Internet protocol information from Layer 3 of the OSI Model). In many situations, this information is processed in conjunction with the routing table (also known as forwarding table). Routers use routing tables to determine what interface to forward packets (this can include the "null" also known as the "black hole" interface because data can go into it, however, no further processing is done for said data).

Firewalls

A firewall is an important aspect of a network with respect to security. It typically rejects access requests from unsafe sources while allowing actions from recognized ones. The vital role firewalls play in network security grows in parallel with the constant increase in 'cyber' attacks for the purpose of stealing/corrupting data, planting viruses, etc.

Network performance

Network performance refers to the service quality of a telecommunications product as seen by the customer. It should not be seen merely as an attempt to get "more through" the network.

The following list gives examples of Network Performance measures for a circuit-switched network and one type of packet-switched network, viz. ATM:

- Circuit-switched networks: In circuit switched networks, network performance is synonymous with the grade of service. The number of rejected calls is a measure of how well the network is performing under heavy traffic loads.[19] Other types of performance measures can include noise, echo and so on.
- ATM: In an Asynchronous Transfer Mode (ATM) network, performance can be measured by line rate, quality of service (QoS), data throughput, connect time, stability, technology, modulation technique and modem enhancements.[20]

There are many different ways to measure the performance of a network, as each network is different in nature and design. Performance can also be modelled instead of measured; one example of this is using state transition diagrams to model queuing performance in a circuit-switched network. These diagrams allow the network planner to analyze how the network will perform in each state, ensuring that the network will be optimally designed.[21]

Network security

In the field of networking, the area of **network security**[22] consists of the provisions and policies adopted by the network administrator to prevent and monitor unauthorized access, misuse, modification, or denial of the computer network and network-accessible resources. Network security is the authorization of access to data in a network, which is controlled by the network administrator. Users are assigned an ID and password that allows them access to information and programs within their authority. Network Security covers a variety of computer networks, both public and private that are used in everyday jobs conducting transactions and communications among businesses, government agencies and individuals.

Network resilience

In computer networking: "**Resilience** is the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation."[23]

Views of networks

Users and network administrators typically have different views of their networks. Users can share printers and some servers from a workgroup, which usually means they are in the same geographic location and are on the same LAN, whereas a Network Administrator is responsible to keep that network up and running. A community of interest has less of a connection of being in a local area, and should be thought of as a set of arbitrarily located users who share a set of servers, and possibly also communicate via peer-to-peer technologies.

Network administrators can see networks from both physical and logical perspectives. The physical perspective involves geographic locations, physical cabling, and the network elements (e.g., routers, bridges and application layer gateways) that interconnect the physical media. Logical networks, called, in the TCP/IP architecture, subnets, map onto one or more physical media. For example, a common practice in a campus of buildings is to make a set of LAN cables in each building appear to be a common subnet, using virtual LAN (VLAN) technology.

Both users and administrators will be aware, to varying extents, of the trust and scope characteristics of a network. Again using TCP/IP architectural terminology, an intranet is a community of interest under private administration usually by an enterprise, and is only accessible by authorized users (e.g. employees).[24] Intranets do not have to be connected to the Internet, but generally have a limited connection. An extranet is an extension of an intranet that allows secure communications to users outside of the intranet (e.g. business partners, customers).[24]

Unofficially, the Internet is the set of users, enterprises, and content providers that are interconnected by Internet Service Providers (ISP). From an engineering viewpoint, the Internet is the set of subnets, and aggregates of subnets, which share the registered IP address space and exchange information about the reachability of those IP addresses using the Border Gateway Protocol. Typically, the human-readable names of servers are translated to IP addresses, transparently to users, via the directory function of the Domain Name System (DNS).

Over the Internet, there can be business-to-business (B2B), business-to-consumer (B2C) and consumer-to-consumer (C2C) communications. Especially when money or sensitive information is exchanged, the communications are apt to be **secured** by some form of communications security mechanism. Intranets and extranets can be securely superimposed onto the Internet, without any access by general Internet users and administrators, using secure Virtual Private Network (VPN) technology.

References

[1] *Computer network definition* (http://www.atis.org/glossary/definition.aspx?id=6555), , retrieved 2011-11-12

[2] Michael A. Banks (2008). *On the way to the web: the secret history of the internet and its founders* (http://books.google.com/books?id=P9wbSjO9WMMC&pg=PA1). Apress. p. 1. ISBN 9781430208693. .

[3] Christos J. P. Moschovitis (1998). *History of the Internet: a chronology, 1843 to the present* (http://books.google.com/?id=Hu5SAAAAMAAJ&dq=intitle:"history+of+the+internet"+sage+sabre&q=sage+sabre's#search_anchor). ABC-CLIO. p. 36. ISBN 9781576071182. .

[4] Chris Sutton. "Internet Began 35 Years Ago at UCLA with First Message Ever Sent Between Two Computers" (http://web.archive.org/web/20080308120314/http://www.engineer.ucla.edu/stories/2004/Internet35.htm). *UCLA*. Archived from the original (http://www.engineer.ucla.edu/stories/2004/Internet35.htm) on March 8, 2008. .

[5] *Broadband Over Powerline* (http://www.arrl.org/broadband-over-powerline-bpl), The National Association for Amateur Radio, , retrieved 2011-11-12

[6] "The Likelihood and Extent of Radio Frequency Interference from In-Home PLT Devices" (http://stakeholders.ofcom.org.uk/binaries/research/technology-research/pltreport.pdf). Ofcom. . Retrieved 18 June 2011.

[7] "Mobile Broadband Wireless connections (MBWA)" (http://grouper.ieee.org/groups/802/20/). . Retrieved 2011-11-12.

[8] Bergen Linux User Group's CPIP Implementation (http://www.blug.linux.no/rfc1149)

[9] A. Hooke (September 2000), *Interplanetary Internet* (http://www.ipnsig.org/reports/ISART9-2000.pdf), Third Annual International Symposium on Advanced Radio Technologies, , retrieved 2011-11-12

[10] Martin, Thomas. "Design Principles for DSL-Based Access Solutions" (http://www.gsi.dit.upm.es/~legf/Varios/XDSL_MARTI.PDF). . Retrieved 18 June 2011.

[11] "personal area network (PAN)" (http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci546288,00.html). . Retrieved January 29, 2011.

[12] *New global standard for fully networked home* (http://www.itu.int/ITU-T/newslog/New+Global+Standard+For+Fully+Networked+Home.aspx), ITU-T, 2008-12-12, , retrieved 2011-11-12

[13] *IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force* (http://www.ieee802.org/3/ba/), , retrieved 2011-11-12

[14] D. Andersen; H. Balakrishnan; M. Kaashoek; R. Morris (10-2001), *Resilient Overlay Networks* (http://nms.lcs.mit.edu/papers/ron-sosp2001.html)*], Association for Computing Machinery, , retrieved 2011-11-12*

[15] http://esm.cs.cmu.edu/

[16] http://standards.ieee.org/regauth/oui/oui.txt

[17] "Define switch." (http://www.webopedia.com/TERM/s/switch.html). www.webopedia.com. . Retrieved April 8, 2008.

[18] "Basic Components of a Local Area Network (LAN)" (http://networkbits.net/lan-components/local-area-network-lan-basic-components/). NetworkBits.net. . Retrieved April 8, 2008.

[19] *Teletraffic Engineering Handbook* (http://web.archive.org/web/20070111015452/http://oldwww.com.dtu.dk/teletraffic/handbook/telenook.pdf), ITU-T Study Group 2, archived from the original (http://www.com.dtu.dk/teletraffic/handbook/telenook.pdf) on 2007-01-11,

[20] Telecommunications Magazine Online (http://www.telecommagazine.com), Americas January 2003, Issue Highlights, Online Exclusive: Broadband Access Maximum Performance, Retrieved on February 13, 2005.

[21] "State Transition Diagrams" (http://cne.gmu.edu/modules/os_perf/std.t.html). . Retrieved July 13, 2003.

[22] Simmonds, A; Sandilands, P; van Ekert, L (2004). "An Ontology for Network Security Attacks". *Lecture Notes in Computer Science*. Lecture Notes in Computer Science **3285**: 317–323. doi:10.1007/978-3-540-30176-9_41. ISBN 978-3-540-23659-7.

[23] "Definitions: Resilience" (http://wiki.ittc.ku.edu/resilinets_wiki/index.php/Definitions#Resilience). ResiliNets Research Initiative. . Retrieved 2011-11-12.

[24] RFC 2547

This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm).

Further reading

- Shelly, Gary, et al. "Discovering Computers" 2003 Edition
- Cisco Systems, Inc., (2003, March 14). CCNA: network media types. Retrieved from ciscopress.com (http://www.ciscopress.com/articles/article.asp?p=31276&rll=1)
- Wendell Odom,Rus Healy, Denise Donohue. (2010) CCIE Routing and Switching. Indianapolis, IN: Cisco Press
- Kurose James F and Keith W. Ross : Computer Networking: A Top-Down Approach Featuring the Internet, Pearson Education 2005.
- Andrew S. Tanenbaum, *Computer Networks*, Fourth Edition, Pearson Education 2006 (ISBN 0-13-349945-6).
- William Stallings, *Computer Networking with Internet Protocols and Technology*, Pearson Education 2004.
- Important publications in computer networks
- Vinton G. Cerf "Software: Global Infrastructure for the 21st Century" (http://www.cs.washington.edu/homes/lazowska/cra/networks.html)
- Meyers, Mike, "Mike Meyers' Certification Passport: Network+" ISBN 0072253487"
- Odom, Wendall, "CCNA Certification Guide"
- Network Communication Architecture and Protocols: OSI Network Architecture 7 Layers Model

External links

- Easy Network Concepts (http://www.netfilter.org/documentation/HOWTO/networking-concepts-HOWTO.html) (Linux kernel specific)
- Computer Networks and Protocol (http://nsgn.net/osi_reference_model/) (Research document, 2006)
- Computer Networking Glossary (http://compnetworking.about.com/od/basicnetworkingconcepts/l/blglossary.htm)
- Networking (http://www.dmoz.org/Computers/Software/Networking//) at the Open Directory Project

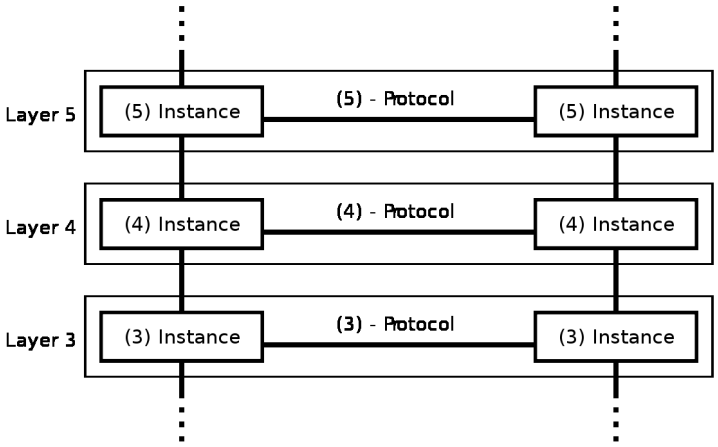
OSI model

The **Open Systems Interconnection model** (**OSI model**) is a product of the Open Systems Interconnection effort at the International Organization for Standardization. It is a prescription of characterizing and standardizing the functions of a communications system in terms of abstraction layers. Similar communication functions are grouped into logical layers. An instance of a layer provides services to its upper layer instances while receiving services from the layer below.

For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of that path. Two instances at one layer are connected by a horizontal connection on that layer.

History

Work on a layered model of network architecture was started and the International Organization for Standardization (ISO) began to develop its OSI framework architecture. OSI had two major components: an *abstract model* of networking, called the Basic Reference Model or seven-layer model, and a set of specific protocols.



Communication in the OSI-Model (example with layers 3 to 5)

The concept of a seven-layer model was provided by the work of Charles Bachman, Honeywell Information Services. Various aspects of OSI design evolved from experiences with the ARPANET, the fledgling Internet, NPLNET, EIN, CYCLADES network and the work in IFIP WG6.1. The new design was documented in ISO 7498 and its various addenda. In this model, a networking system was divided into layers. Within each layer, one or more entities implement its functionality. Each entity interacted directly only with the layer immediately beneath it, and provided facilities for use by the layer above it.

Protocols enabled an entity in one host to interact with a corresponding entity at the same layer in another host. Service definitions abstractly described the functionality provided to an (N)-layer by an (N-1) layer, where N was one of the seven layers of protocols operating in the local host.

The OSI standards documents are available from the ITU-T as the X.200-series of recommendations.[1] Some of the protocol specifications were also available as part of the ITU-T X series. The equivalent ISO and ISO/IEC standards for the OSI model were available from ISO, but only some of them without fees.[2]

Description of OSI layers

According to recommendation X.200, there are seven layers, each generically known as an N layer. An N+1 entity requests services from the layer-N entity.

At each level, two entities (N-entity peers) interact by means of the N protocol by transmitting protocol data units (PDU).

A Service Data Unit (SDU) is a specific unit of data that has been passed down from an OSI layer to a lower layer, and which the lower layer has not yet encapsulated into a protocol data unit (PDU). An SDU is a set of data that is sent by a user of the services of a given layer, and is transmitted semantically unchanged to a peer service user.

The PDU at any given layer, layer N, is the SDU of the layer below, layer N-1. In effect the SDU is the 'payload' of a given PDU. That is, the process of changing a SDU to a PDU, consists of an encapsulation process, performed by the lower layer. All the data contained in the SDU becomes encapsulated within the PDU. The layer N-1 adds headers or footers, or both, to the SDU, transforming it into a PDU of layer N-1. The added headers or footers are part of the process used to make it possible to get data from a source to a destination.

OSI Model			
	Data unit	**Layer**	**Function**
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	End-to-end connections, reliability and flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Some orthogonal aspects, such as management and security, involve every layer.

Security services are not related to a specific layer: they can be related by a number of layers, as defined by ITU-T X.800 Recommendation.[3]

These services are aimed to improve the CIA triad (confidentiality, integrity, and availability) of transmitted data. Actually the availability of communication service is determined by network design and/or network management protocols. Appropriate choices for these are needed to protect against denial of service.

Layer 1: physical layer

The physical layer defines electrical and physical specifications for devices. In particular, it defines the relationship between a device and a transmission medium, such as a copper or optical cable. This includes the layout of pins, voltages, cable specifications, hubs, repeaters, network adapters, host bus adapters (HBA used in storage area networks) and more.

The major functions and services performed by the physical layer are:

- Establishment and termination of a connection to a communications medium.
- Participation in the process whereby the communication resources are effectively shared among multiple users. For example, contention resolution and flow control.
- Modulation, or conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. These are signals operating over the physical cabling (such as copper and optical fiber) or over a radio link.

Parallel SCSI buses operate in this layer, although it must be remembered that the logical SCSI protocol is a transport layer protocol that runs over this bus. Various physical-layer Ethernet standards are also in this layer; Ethernet incorporates both this layer and the data link layer. The same applies to other local-area networks, such as token ring, FDDI, ITU-T G.hn and IEEE 802.11, as well as personal area networks such as Bluetooth and IEEE 802.15.4.

Layer 2: data link layer

The data link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the physical layer. Originally, this layer was intended for point-to-point and point-to-multipoint media, characteristic of wide area media in the telephone system. Local area network architecture, which included broadcast-capable multiaccess media, was developed independently of the ISO work in IEEE Project 802. IEEE work assumed sublayering and management functions not required for WAN use. In modern practice, only error detection, not flow control using sliding window, is present in data link protocols such as Point-to-Point Protocol (PPP), and, on local area networks, the IEEE 802.2 LLC layer is not used for most protocols on the Ethernet, and on other local area networks, its flow control and acknowledgment mechanisms are rarely used. Sliding window flow control and acknowledgment is used at the transport layer by protocols such as TCP, but is still used in niches where X.25 offers performance advantages.

The ITU-T G.hn standard, which provides high-speed local area networking over existing wires (power lines, phone lines and coaxial cables), includes a complete data link layer which provides both error correction and flow control by means of a selective repeat Sliding Window Protocol.

Both WAN and LAN service arrange bits, from the physical layer, into logical sequences called frames. Not all physical layer bits necessarily go into frames, as some of these bits are purely intended for physical layer functions. For example, every fifth bit of the FDDI bit stream is not used by the layer.

WAN protocol architecture

Connection-oriented WAN data link protocols, in addition to framing, detect and may correct errors. They are also capable of controlling the rate of transmission. A WAN data link layer might implement a sliding window flow control and acknowledgment mechanism to provide reliable delivery of frames; that is the case for Synchronous Data Link Control (SDLC) and HDLC, and derivatives of HDLC such as LAPB and LAPD.

IEEE 802 LAN architecture

Practical, connectionless LANs began with the pre-IEEE Ethernet specification, which is the ancestor of IEEE 802.3. This layer manages the interaction of devices with a shared medium, which is the function of a media access control (MAC) sublayer. Above this MAC sublayer is the media-independent IEEE 802.2 Logical Link Control (LLC) sublayer, which deals with addressing and multiplexing on multiaccess media.

While IEEE 802.3 is the dominant wired LAN protocol and IEEE 802.11 the wireless LAN protocol, obsolescent MAC layers include Token Ring and FDDI. The MAC sublayer detects but does not correct errors.

Layer 3: network layer

The network layer provides the functional and procedural means of transferring variable length data sequences from a source host on one network to a destination host on a different network, while maintaining the quality of service requested by the transport layer (in contrast to the data link layer which connects hosts within the same network). The network layer performs network routing functions, and might also perform fragmentation and reassembly, and report delivery errors. Routers operate at this layer, sending data throughout the extended network and making the Internet possible. This is a logical addressing scheme – values are chosen by the network engineer. The addressing scheme is not hierarchical.

The network layer may be divided into three sublayers:

1. Subnetwork access – that considers protocols that deal with the interface to networks, such as X.25;
2. Subnetwork-dependent convergence – when it is necessary to bring the level of a transit network up to the level of networks on either side
3. Subnetwork-independent convergence – handles transfer across multiple networks.

An example of this latter case is CLNP, or IPv7 ISO 8473. It manages the connectionless transfer of data one hop at a time, from end system to ingress router, router to router, and from egress router to destination end system. It is not responsible for reliable delivery to a next hop, but only for the detection of erroneous packets so they may be discarded. In this scheme, IPv4 and IPv6 would have to be classed with X.25 as subnet access protocols because they carry interface addresses rather than node addresses.

A number of layer-management protocols, a function defined in the Management Annex, ISO 7498/4, belong to the network layer. These include routing protocols, multicast group management, network-layer information and error, and network-layer address assignment. It is the function of the payload that makes these belong to the network layer, not the protocol that carries them.

Layer 4: transport layer

The transport layer provides transparent transfer of data between end users, providing reliable data transfer services to the upper layers. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state- and connection-oriented. This means that the transport layer can keep track of the segments and retransmit those that fail. The transport layer also provides the acknowledgement of the successful data transmission and sends the next data if no errors occurred.

OSI defines five classes of connection-mode transport protocols ranging from class 0 (which is also known as TP0 and provides the least features) to class 4 (TP4, designed for less reliable networks, similar to the Internet). Class 0 contains no error recovery, and was designed for use on network layers that provide error-free connections. Class 4 is closest to TCP, although TCP contains functions, such as the graceful close, which OSI assigns to the session layer. Also, all OSI TP connection-mode protocol classes provide expedited data and preservation of record boundaries. Detailed characteristics of TP0-4 classes are shown in the following table:[4]

Feature Name	TP0	TP1	TP2	TP3	TP4
Connection oriented network	Yes	Yes	Yes	Yes	Yes
Connectionless network	No	No	No	No	Yes
Concatenation and separation	No	Yes	Yes	Yes	Yes
Segmentation and reassembly	Yes	Yes	Yes	Yes	Yes
Error Recovery	No	Yes	Yes	Yes	Yes
Reinitiate connection (if an excessive number of PDUs are unacknowledged)	No	Yes	No	Yes	No
Multiplexing and demultiplexing over a single virtual circuit	No	No	Yes	Yes	Yes
Explicit flow control	No	No	Yes	Yes	Yes
Retransmission on timeout	No	No	No	No	Yes
Reliable Transport Service	No	Yes	No	Yes	Yes

Perhaps an easy way to visualize the transport layer is to compare it with a Post Office, which deals with the dispatch and classification of mail and parcels sent. Do remember, however, that a post office manages the outer envelope of mail. Higher layers may have the equivalent of double envelopes, such as cryptographic presentation services that can be read by the addressee only. Roughly speaking, tunneling protocols operate at the transport layer, such as carrying non-IP protocols such as IBM's SNA or Novell's IPX over an IP network, or end-to-end encryption with

IPsec. While Generic Routing Encapsulation (GRE) might seem to be a network-layer protocol, if the encapsulation of the payload takes place only at endpoint, GRE becomes closer to a transport protocol that uses IP headers but contains complete frames or packets to deliver to an endpoint. L2TP carries PPP frames inside transport packet.

Although not developed under the OSI Reference Model and not strictly conforming to the OSI definition of the transport layer, the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) of the Internet Protocol Suite are commonly categorized as layer-4 protocols within OSI.

Layer 5: session layer

The session layer controls the dialogues (connections) between computers. It establishes, manages and terminates the connections between the local and remote application. It provides for full-duplex, half-duplex, or simplex operation, and establishes checkpointing, adjournment, termination, and restart procedures. The OSI model made this layer responsible for graceful close of sessions, which is a property of the Transmission Control Protocol, and also for session checkpointing and recovery, which is not usually used in the Internet Protocol Suite. The session layer is commonly implemented explicitly in application environments that use remote procedure calls.

Layer 6: presentation layer

The presentation layer establishes context between application-layer entities, in which the higher-layer entities may use different syntax and semantics if the presentation service provides a mapping between them. If a mapping is available, presentation service data units are encapsulated into session protocol data units, and passed down the stack.

This layer provides independence from data representation (e.g., encryption) by translating between application and network formats. The presentation layer transforms data into the form that the application accepts. This layer formats and encrypts data to be sent across a network. It is sometimes called the syntax layer.[5]

The original presentation structure used the basic encoding rules of Abstract Syntax Notation One (ASN.1), with capabilities such as converting an EBCDIC-coded text file to an ASCII-coded file, or serialization of objects and other data structures from and to XML.

Layer 7: application layer

The application layer is the OSI layer closest to the end user, which means that both the OSI application layer and the user interact directly with the software application. This layer interacts with software applications that implement a communicating component. Such application programs fall outside the scope of the OSI model. Application-layer functions typically include identifying communication partners, determining resource availability, and synchronizing communication. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application with data to transmit. When determining resource availability, the application layer must decide whether sufficient network or the requested communication exist. In synchronizing communication, all communication between applications requires cooperation that is managed by the application layer. Some examples of application-layer implementations also include:

- On OSI stack:
  - FTAM File Transfer and Access Management Protocol
  - X.400 Mail
  - Common management information protocol (CMIP)
- On TCP/IP stack:
  - Hypertext Transfer Protocol (HTTP),
  - File Transfer Protocol (FTP),
  - Simple Mail Transfer Protocol (SMTP)

- Simple Network Management Protocol (SNMP).

Cross-layer functions

There are some functions or services that are not tied to a given layer, but they can affect more than one layer. Examples are

- security service (telecommunication)[3] as defined by ITU-T X.800 Recommendation.
- management functions, i.e. functions that permit to configure, instantiate, monitor, terminate the communications of two or more entities: there is a specific application layer protocol, common management information protocol (CMIP) and its corresponding service, common management information service (CMIS), they need to interact with every layer in order to deal with their instances.
- MPLS operates at an OSI-model layer that is generally considered to lie between traditional definitions of layer 2 (data link layer) and layer 3 (network layer), and thus is often referred to as a "layer-2.5" protocol. It was designed to provide a unified data-carrying service for both circuit-based clients and packet-switching clients which provide a datagram service model. It can be used to carry many different kinds of traffic, including IP packets, as well as native ATM, SONET, and Ethernet frames.
- ARP is used to translate IPv4 addresses (OSI layer 3) into Ethernet MAC addresses (OSI layer 2).

Interfaces

Neither the OSI Reference Model nor OSI protocols specify any programming interfaces, other than as deliberately abstract service specifications. Protocol specifications precisely define the interfaces between different computers, but the software interfaces inside computers are implementation-specific.

For example Microsoft Windows' Winsock, and Unix's Berkeley sockets and System V Transport Layer Interface, are interfaces between applications (layer 5 and above) and the transport (layer 4). NDIS and ODI are interfaces between the media (layer 2) and the network protocol (layer 3).

Interface standards, except for the physical layer to media, are approximate implementations of OSI service specifications.

Examples

Layer		OSI protocols	TCP/IP protocols	Signaling System 7[6]	AppleTalk	IPX	SNA	UMTS	Misc. examples
#	Name								
7	Application	FTAM, X.400, X.500, DAP, ROSE, RTSE, ACSE[7] CMIP[8]	NNTP, SIP, SSI, DNS, FTP, Gopher, HTTP, NFS, NTP, DHCP, SMPP, SMTP, SNMP, Telnet, RIP, BGP	INAP, MAP, TCAP, ISUP, TUP	AFP, ZIP, RTMP, NBP	RIP, SAP	APPC		HL7, Modbus
6	Presentation	ISO/IEC 8823, X.226, ISO/IEC 9576-1, X.236	MIME, SSL, TLS, XDR		AFP				TDI, ASCII, EBCDIC, MIDI, MPEG

5	Session	ISO/IEC 8327, X.225, ISO/IEC 9548-1, X.235	Sockets. Session establishment in TCP, RTP		ASP, ADSP, PAP	NWLink	DLC?		Named pipes, NetBIOS, SAP, half duplex, full duplex, simplex, RPC
4	Transport	ISO/IEC 8073, TP0, TP1, TP2, TP3, TP4 (X.224), ISO/IEC 8602, X.234	TCP, UDP, SCTP, DCCP			DDP, SPX			NBF
3	Network	ISO/IEC 8208, X.25 (PLP), ISO/IEC 8878, X.223, ISO/IEC 8473-1, CLNP X.233.	IP, IPsec, ICMP, IGMP, OSPF	SCCP, MTP	ATP (TokenTalk or EtherTalk)	IPX		RRC (Radio Resource Control) Packet Data Convergence Protocol (PDCP) and BMC (Broadcast/Multicast Control)	NBF, Q.931, IS-IS
2	Data Link	ISO/IEC 7666, X.25 (LAPB), Token Bus, X.222, ISO/IEC 8802-2 LLC Type 1 and 2[9]	PPP, SBTV SLIP, PPTP	MTP, Q.710	LocalTalk, AppleTalk Remote Access, PPP	IEEE 802.3 framing, Ethernet II framing	SDLC	LLC (Logical Link Control), MAC (Media Access Control)	802.3 (Ethernet), 802.11a/b/g/n MAC/LLC, 802.1Q (VLAN), ATM, HDP, FDDI, Fibre Channel, Frame Relay, HDLC, ISL, PPP, Q.921, Token Ring, CDP, NDP ARP (maps layer 3 to layer 2 address), ITU-T G.hn DLL CRC, Bit stuffing, ARQ, Data Over Cable Service Interface Specification (DOCSIS), interface bonding
1	Physical	X.25 (X.21bis, EIA/TIA-232, EIA/TIA-449, EIA-530, G.703)[9]		MTP, Q.710	RS-232, RS-422, STP, PhoneNet		Twinax	UMTS Physical layer or L1	RS-232, Full duplex, RJ45, V.35, V.34, I.430, I.431, T1, E1, 10BASE-T, 100BASE-TX, POTS, SONET, SDH, DSL, 802.11a/b/g/n PHY, ITU-T G.hn PHY, Controller Area Network, Data Over Cable Service Interface Specification (DOCSIS)

Comparison with TCP/IP

In the TCP/IP model of the Internet, protocols are deliberately not as rigidly designed into strict layers as in the OSI model.[10] RFC 3439 contains a section entitled "Layering considered harmful." However, TCP/IP does recognize four broad layers of functionality which are derived from the operating scope of their contained protocols, namely the scope of the software application, the end-to-end transport connection, the internetworking range, and lastly the scope of the direct links to other nodes on the local network.

Even though the concept is different from the OSI model, these layers are nevertheless often compared with the OSI layering scheme in the following way: The Internet application layer includes the OSI application layer, presentation layer, and most of the session layer. Its end-to-end transport layer includes the graceful close function of the OSI session layer as well as the OSI transport layer. The internetworking layer (Internet layer) is a subset of the OSI network layer (see above), while the link layer includes the OSI data link and physical layers, as well as parts of OSI's network layer. These comparisons are based on the original seven-layer protocol model as defined in ISO 7498, rather than refinements in such things as the internal organization of the network layer document.

The presumably strict peer layering of the OSI model as it is usually described does not present contradictions in TCP/IP, as it is permissible that protocol usage does not follow the hierarchy implied in a layered model. Such examples exist in some routing protocols (e.g., OSPF), or in the description of tunneling protocols, which provide a link layer for an application, although the tunnel host protocol may well be a transport or even an application layer protocol in its own right.

References

[1] ITU-T X-Series Recommendations (http://www.itu.int/rec/T-REC-X/en)
[2] "Publicly Available Standards" (http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html). Standards.iso.org. 2010-07-30. . Retrieved 2010-09-11.
[3] X.800 : Security architecture for Open Systems Interconnection for CCITT applications (http://www.itu.int/rec/T-REC-X.800-199103-I/e)
[4] "ITU-T Recommendation X.224 (11/1995) ISO/IEC 8073" (http://www.itu.int/rec/T-REC-X.224-199511-I/en/). .
[5] Grigonis, Richard (2000). *Computer telephony encyclopedia* (http://books.google.com/books?id=cUYk0ZhOxpEC&printsec=frontcover&dq=computer+telephony+encyclopedia&ct=result#v=onepage&q&f=false). CMP. pp. 331. .
[6] ITU-T Recommendation Q.1400 (03/1993) (http://www.itu.int/rec/T-REC-Q.1400/en/), *Architecture framework for the development of signaling and OA&M protocols using OSI concepts*, pp 4, 7.
[7] ITU Rec. X.227 (ISO 8650), X.217 (ISO 8649)
[8] X.700 series of recommendations from the ITU-T (in particular X.711), and ISO 9596
[9] CISCO Cisco Systems, Inc. Internetworking Technology Handbook OSI Model Physical layer (http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-Internet.html#wp1020669)
[10] RFC 3439

External links

- ISO/IEC standard 7498-1:1994 (http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip) (PDF document inside ZIP archive) (requires HTTP cookies in order to accept licence agreement)
- ITU-T X.200 (the same contents as from ISO) (http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items)
- The ISO OSI Reference Model , Beluga graph of data units and groups of layers (http://infchg.appspot.com/usr?at=1263939371)
- OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.9497&rep=rep1&type=pdf)PDF (776 KB), Hubert Zimmermann, IEEE Transactions on Communications, vol. 28, no. 4, April 1980, pp. 425 – 432.

- Cisco Systems Internetworking Technology Handbook (http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook)

TCP/IP model

The **TCP/IP model** (Transmission Control Protocol/Internet Protocol) is a descriptive framework for the Internet Protocol Suite of computer network protocols created in the 1970s by DARPA, an agency of the United States Department of Defense. It evolved from ARPANET, which was an early wide area network and a predecessor of the Internet. The TCP/IP Model is sometimes called the *Internet Model* or less often the *DoD Model.*

The TCP/IP model describes a set of general design guidelines and implementations of specific networking protocols to enable computers to communicate over a network. TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. Protocols exist for a variety of different types of communication services between computers.

TCP/IP has four abstraction layers as defined in RFC 1122. This layer architecture is often compared with the seven-layer OSI Reference Model; using terms such as *Internet reference model*, incorrectly, however, because it is descriptive while the OSI Reference Model was intended to be prescriptive, hence being a reference model.

The TCP/IP model and related protocols are maintained by the Internet Engineering Task Force (IETF).

Key architectural principles

An early architectural document, RFC 1122, emphasizes architectural principles over layering.[1]

- End-to-End Principle: This principle has evolved over time. Its original expression put the maintenance of state and overall intelligence at the edges, and assumed the Internet that connected the edges retained no state and concentrated on speed and simplicity. Real-world needs for firewalls, network address translators, web content caches and the like have forced changes in this principle.[2]
- Robustness Principle: "In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear)." [3] "The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features." [4]

Even when the layers are examined, the assorted architectural documents—there is no single architectural model such as ISO 7498, the OSI reference model—have fewer and less rigidly-defined layers than the OSI model, and thus provide an easier fit for real-world protocols. In point of fact, one frequently referenced document, RFC 1958, does not contain a stack of layers. The lack of emphasis on layering is a strong difference between the IETF and OSI approaches. It only refers to the existence of the "internetworking layer" and generally to "upper layers"; this document was intended as a 1996 "snapshot" of the architecture: "The Internet and its architecture have grown in evolutionary fashion from modest beginnings, rather than from a Grand Plan. While this process of evolution is one of the main reasons for the technology's success, it nevertheless seems useful to record a snapshot of the current principles of the Internet architecture."

RFC 1122, entitled *Host Requirements*, is structured in paragraphs referring to layers, but the document refers to many other architectural principles not emphasizing layering. It loosely defines a four-layer model, with the layers having names, not numbers, as follows:

- Application Layer (process-to-process): This is the scope within which applications create user data and communicate this data to other processes or applications on another or the same host. The communications partners are often called *peers*. This is where the "higher level" protocols such as SMTP, FTP, SSH, HTTP, etc.

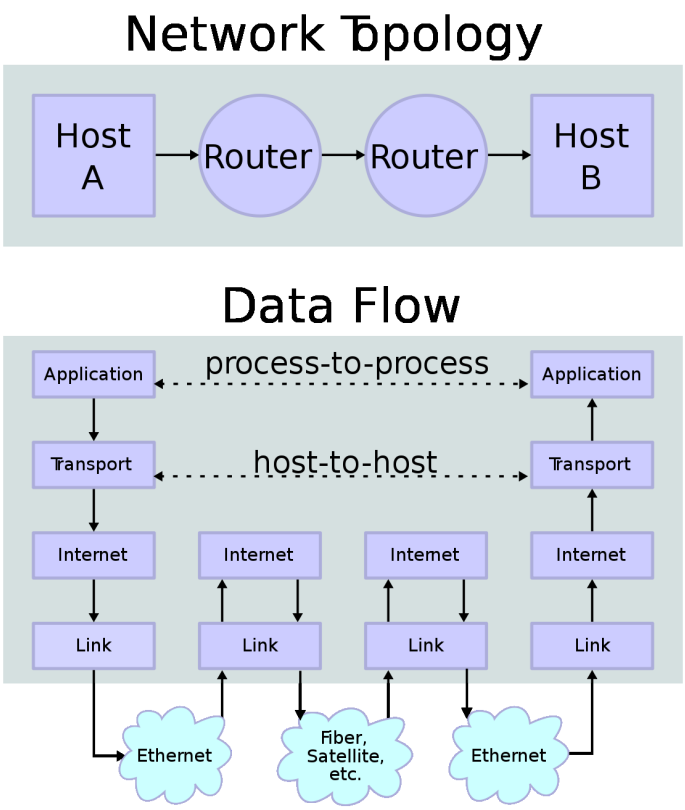
operate.

- Transport Layer (host-to-host): The Transport Layer constitutes the networking regime between two network hosts, either on the local network or on remote networks separated by routers. The Transport Layer provides a uniform networking interface that hides the actual topology (layout) of the underlying network connections. This is where flow-control, error-correction, and connection protocols exist, such as TCP. This layer deals with opening and maintaining connections between Internet hosts.
- Internet Layer (internetworking): The Internet Layer has the task of exchanging datagrams across network boundaries. It is therefore also referred to as the layer that establishes internetworking, indeed, it defines and establishes the Internet. This layer defines the addressing and routing structures used for the TCP/IP protocol suite. The primary protocol in this scope is the Internet Protocol, which defines IP addresses. Its function in routing is to transport datagrams to the next IP router that has the connectivity to a network closer to the final data destination.
- Link Layer: This layer defines the networking methods within the scope of the local network link on which hosts communicate without intervening routers. This layer describes the protocols used to describe the local network topology and the interfaces needed to affect transmission of Internet Layer datagrams to next-neighbor hosts. (cf. the OSI Data Link Layer).

The Internet Protocol Suite and the layered protocol stack design were in use before the OSI model was established. Since then, the TCP/IP model has been compared with the OSI model in books and classrooms, which often results in confusion because the two models use different assumptions, including about the relative importance of strict layering.

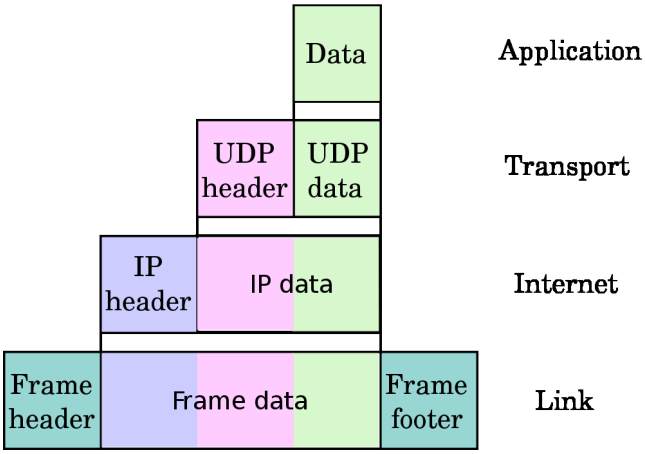
Layers in the TCP/IP model

The layers near the top are logically closer to the user application, while those near the bottom are logically closer to the physical transmission of the data. Viewing layers as providing or consuming a service is a method of abstraction to isolate upper layer protocols from the nitty-gritty detail of transmitting bits over, for example, Ethernet and collision detection, while the lower layers avoid having to know the details of each and every application and its protocol.



Two Internet hosts connected via two routers and the corresponding layers used at each hop.

This abstraction also allows upper layers to provide services that the lower layers cannot, or choose not to, provide. Again, the original OSI Reference Model was extended to include connectionless services (OSIRM CL).[5] For example, IP is not designed to be reliable and is a best effort delivery protocol. This means that all transport layer implementations must choose whether or not to provide reliability and to what degree. UDP provides data integrity (via a checksum) but does not guarantee delivery; TCP provides both data integrity and delivery guarantee (by retransmitting until the receiver acknowledges the reception of the packet).



Encapsulation of application data descending through the TCP/IP layers

This model lacks the formalism of the OSI reference model and associated documents, but the IETF does not use a formal model and does not consider this a limitation, as in the comment by David D. Clark, "We reject: kings, presidents and voting. We believe in: rough consensus and running code." Criticisms of this model, which have been made with respect to the OSI Reference Model, often do not consider ISO's later extensions to that model.

1. For multiaccess links with their own addressing systems (e.g. Ethernet) an address mapping protocol is needed. Such protocols can be considered to be below IP but above the existing link system. While the IETF does not use the terminology, this is a subnetwork dependent convergence facility according to an extension to the OSI model,

the Internal Organization of the Network Layer (IONL).[6]

2. ICMP & IGMP operate on top of IP but do not transport data like UDP or TCP. Again, this functionality exists as layer management extensions to the OSI model, in its *Management Framework* (OSIRM MF) [7]
3. The SSL/TLS library operates above the transport layer (uses TCP) but below application protocols. Again, there was no intention, on the part of the designers of these protocols, to comply with OSI architecture.
4. The link is treated like a black box here. This is fine for discussing IP (since the whole point of IP is it will run over virtually anything). The IETF explicitly does not intend to discuss transmission systems, which is a less academic but practical alternative to the OSI Reference Model.

The following is a description of each layer in the TCP/IP networking model starting from the lowest level.

Link Layer

The Link Layer (or Network Access Layer) is the networking scope of the local network connection to which a host is attached. This regime is called the *link* in Internet literature. This is the lowest component layer of the Internet protocols, as TCP/IP is designed to be hardware independent. As a result TCP/IP is able to be implemented on top of virtually any hardware networking technology.

The Link Layer is used to move packets between the Internet Layer interfaces of two different hosts on the same link. The processes of transmitting and receiving packets on a given link can be controlled both in the software device driver for the network card, as well as on firmware or specialized chipsets. These will perform data link functions such as adding a packet header to prepare it for transmission, then actually transmit the frame over a physical medium. The TCP/IP model includes specifications of translating the network addressing methods used in the Internet Protocol to data link addressing, such as Media Access Control (MAC), however all other aspects below that level are implicitly assumed to exist in the Link Layer, but are not explicitly defined.

This is also the layer where packets may be selected to be sent over a virtual private network or other networking tunnel. In this scenario, the Link Layer data may be considered application data which traverses another instantiation of the IP stack for transmission or reception over another IP connection. Such a connection, or virtual link, may be established with a transport protocol or even an application scope protocol that serves as a tunnel in the Link Layer of the protocol stack. Thus, the TCP/IP model does not dictate a strict hierarchical encapsulation sequence.

Internet Layer

The Internet Layer solves the problem of sending packets across one or more networks. Internetworking requires sending data from the source network to the destination network. This process is called routing.[8]

In the Internet Protocol Suite, the Internet Protocol performs two basic functions:

- *Host addressing and identification*: This is accomplished with a hierarchical addressing system (see IP address).
- *Packet routing*: This is the basic task of getting packets of data (datagrams) from source to destination by sending them to the next network node (router) closer to the final destination.

IP can carry data for a number of different upper layer protocols. These protocols are each identified by a unique protocol number: for example, Internet Control Message Protocol (ICMP) and Internet Group Management Protocol (IGMP) are protocols 1 and 2, respectively.

Some of the protocols carried by IP, such as ICMP (used to transmit diagnostic information about IP transmission) and IGMP (used to manage IP Multicast data) are layered on top of IP but perform internetworking functions. This illustrates the differences in the architecture of the TCP/IP stack of the Internet and the OSI model.

Transport Layer

The Transport Layer's responsibilities include end-to-end message transfer capabilities independent of the underlying network, along with error control, segmentation, flow control, congestion control, and application addressing (port numbers). End to end message transmission or connecting applications at the transport layer can be categorized as either connection-oriented, implemented in Transmission Control Protocol (TCP), or connectionless, implemented in User Datagram Protocol (UDP).

The Transport Layer can be thought of as a transport mechanism, e.g., a vehicle with the responsibility to make sure that its contents (passengers/goods) reach their destination safely and soundly, unless another protocol layer is responsible for safe delivery.

The Transport Layer provides this service of connecting applications through the use of service ports. Since IP provides only a best effort delivery, the Transport Layer is the first layer of the TCP/IP stack to offer reliability. IP can run over a reliable data link protocol such as the High-Level Data Link Control (HDLC). Protocols above transport, such as RPC, also can provide reliability.

For example, the Transmission Control Protocol (TCP) is a connection-oriented protocol that addresses numerous reliability issues to provide a reliable byte stream:

- data arrives in-order
- data has minimal error (i.e. correctness)
- duplicate data is discarded
- lost/discarded packets are resent
- includes traffic congestion control

The newer Stream Control Transmission Protocol (SCTP) is also a reliable, connection-oriented transport mechanism. It is Message-stream-oriented — not byte-stream-oriented like TCP — and provides multiple streams multiplexed over a single connection. It also provides multi-homing support, in which a connection end can be represented by multiple IP addresses (representing multiple physical interfaces), such that if one fails, the connection is not interrupted. It was developed initially for telephony applications (to transport SS7 over IP), but can also be used for other applications.

User Datagram Protocol is a connectionless datagram protocol. Like IP, it is a best effort, "unreliable" protocol. Reliability is addressed through error detection using a weak checksum algorithm. UDP is typically used for applications such as streaming media (audio, video, Voice over IP etc.) where on-time arrival is more important than reliability, or for simple query/response applications like DNS lookups, where the overhead of setting up a reliable connection is disproportionately large. Real-time Transport Protocol (RTP) is a datagram protocol that is designed for real-time data such as streaming audio and video.

TCP and UDP are used to carry an assortment of higher-level applications. The appropriate transport protocol is chosen based on the higher-layer protocol application. For example, the File Transfer Protocol expects a reliable connection, but the Network File System (NFS) assumes that the subordinate Remote Procedure Call protocol, not transport, will guarantee reliable transfer. Other applications, such as VoIP, can tolerate some loss of packets, but not the reordering or delay that could be caused by retransmission.

The applications at any given network address are distinguished by their TCP or UDP port. By convention certain *well known ports* are associated with specific applications. (*See List of TCP and UDP port numbers.*)

Application Layer

The Application Layer refers to the higher-level protocols used by most applications for network communication. Examples of application layer protocols include the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP).[9] Data coded according to application layer protocols are then encapsulated into one or (occasionally) more transport layer protocols (such as the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)), which in turn use lower layer protocols to effect actual data transfer.

Since the IP stack defines no layers between the application and transport layers, the application layer must include any protocols that act like the OSI's presentation and session layer protocols. This is usually done through libraries.

Application Layer protocols generally treat the transport layer (and lower) protocols as "black boxes" which provide a stable network connection across which to communicate, although the applications are usually aware of key qualities of the transport layer connection such as the end point IP addresses and port numbers. As noted above, layers are not necessarily clearly defined in the Internet protocol suite. Application layer protocols are most often associated with client–server applications, and the commoner servers have specific ports assigned to them by the IANA: HTTP has port 80; Telnet has port 23; etc. Clients, on the other hand, tend to use ephemeral ports, i.e. port numbers assigned at random from a range set aside for the purpose.

Transport and lower level layers are largely unconcerned with the specifics of application layer protocols. Routers and switches do not typically "look inside" the encapsulated traffic to see what kind of application protocol it represents, rather they just provide a conduit for it. However, some firewall and bandwidth throttling applications do try to determine what's inside, as with the Resource Reservation Protocol (RSVP). It's also sometimes necessary for Network Address Translation (NAT) facilities to take account of the needs of particular application layer protocols. (NAT allows hosts on private networks to communicate with the outside world via a single visible IP address using port forwarding, and is an almost ubiquitous feature of modern domestic broadband routers).

Hardware and software implementation

Normally, application programmers are concerned only with interfaces in the Application Layer and often also in the Transport Layer, while the layers below are services provided by the TCP/IP stack in the operating system. Microcontroller firmware in the network adapter typically handles link issues, supported by driver software in the operational system. Non-programmable analog and digital electronics are normally in charge of the physical components in the Link Layer, typically using an application-specific integrated circuit (ASIC) chipset for each network interface or other physical standard.

However, hardware or software implementation is not stated in the protocols or the layered reference model. High-performance routers are to a large extent based on fast non-programmable digital electronics, carrying out link level switching.

OSI and TCP/IP layering differences

The three top layers in the OSI model—the Application Layer, the Presentation Layer and the Session Layer—are not distinguished separately in the TCP/IP model where it is just the Application Layer. While some pure OSI protocol applications, such as X.400, also combined them, there is no *requirement* that a TCP/IP protocol stack needs to impose monolithic architecture above the Transport Layer. For example, the Network File System (NFS) application protocol runs over the eXternal Data Representation (XDR) presentation protocol, which, in turn, runs over a protocol with Session Layer functionality, Remote Procedure Call (RPC). RPC provides reliable record transmission, so it can run safely over the best-effort User Datagram Protocol (UDP) transport.

The Session Layer roughly corresponds to the Telnet virtual terminal functionality, which is part of text based protocols such as the HTTP and SMTP TCP/IP model Application Layer protocols. It also corresponds to TCP and UDP port numbering, which is considered as part of the transport layer in the TCP/IP model. Some functions that

would have been performed by an OSI presentation layer are realized at the Internet application layer using the MIME standard, which is used in application layer protocols such as HTTP and SMTP.

Since the IETF protocol development effort is not concerned with strict layering, some of its protocols may not appear to fit cleanly into the OSI model. These conflicts, however, are more frequent when one only looks at the original OSI model, ISO 7498, without looking at the annexes to this model (e.g., ISO 7498/4 Management Framework), or the ISO 8648 Internal Organization of the Network Layer (IONL). When the IONL and Management Framework documents are considered, the ICMP and IGMP are neatly defined as layer management protocols for the network layer. In like manner, the IONL provides a structure for "subnetwork dependent convergence facilities" such as ARP and RARP.

IETF protocols can be encapsulated recursively, as demonstrated by tunneling protocols such as Generic Routing Encapsulation (GRE). While basic OSI documents do not consider tunneling, there is some concept of tunneling in yet another extension to the OSI architecture, specifically the transport layer gateways within the International Standardized Profile framework.[10] The associated OSI development effort, however, has been abandoned given the overwhelming adoption of TCP/IP protocols.

Layer names and number of layers in the literature

The following table shows the layer names and the number of layers of networking models presented in RFCs and textbooks in widespread use in today's university computer networking courses. An evolution can be observed from the original three and four layer models towards a five layer reference model.

Kurose,[11] Forouzan [12]	**Comer,[13] Kozierok[14]**	**Stallings[15]**	**Tanenbaum[16]**	**RFC 1122, Internet STD 3 (1989)**	**Cisco Academy[17]**	**Mike Padlipsky's 1982 "Arpanet Reference Model" (RFC 871)**	**OSI model**
Five layers	*Four+one layers*	*Five layers*	*Five layers*	*Four layers*	*Four layers*	*Three layers*	*Seven layers*
"Five-layer Internet model" or "TCP/IP protocol suite"	"TCP/IP 5-layer reference model"	"TCP/IP model"	"TCP/IP 5-layer reference model"	"Internet model"	"Internet model"	"Arpanet reference model"	ISO model
Application	Application	Application	Application	Application	Application	Application/Process	Application
							Presentation
							Session
Transport	Transport	Host-to-host or transport	Transport	Transport	Transport	Host-to-host	Transport
Network	Internet	Internet	Internet	Internet	Internetwork		Network
Data link	Data link (Network interface)	Network access	Data link	Link	Network interface	Network interface	Data link
Physical	(Hardware)	Physical	Physical				Physical

These textbooks are secondary sources that may contravene the intent of RFC 1122 and other IETF primary sources such as RFC 3439.[18]

Different authors have interpreted the RFCs differently regarding the question whether the Link Layer (and the TCP/IP model) covers Physical Layer issues, or if a hardware layer is assumed below the Link Layer. Some authors have tried to use other names for the Link Layer, such as *network interface layer*, in view to avoid confusion with the Data Link Layer of the seven layer OSI model. Others have attempted to map the Internet Protocol model onto the OSI Model. The mapping often results in a model with five layers where the Link Layer is split into a Data Link

Layer on top of a Physical Layer. In literature with a bottom-up approach to Internet communication,[12] [13] [15] in which hardware issues are emphasized, those are often discussed in terms of physical layer and data link layer.

The Internet Layer is usually directly mapped into the OSI Model's Network Layer, a more general concept of network functionality. The Transport Layer of the TCP/IP model, sometimes also described as the host-to-host layer, is mapped to OSI Layer 4 (Transport Layer), sometimes also including aspects of OSI Layer 5 (Session Layer) functionality. OSI's Application Layer, Presentation Layer, and the remaining functionality of the Session Layer are collapsed into TCP/IP's Application Layer. The argument is that these OSI layers do usually not exist as separate processes and protocols in Internet applications.

However, the Internet protocol stack has never been altered by the Internet Engineering Task Force from the four layers defined in RFC 1122. The IETF makes no effort to follow the OSI model although RFCs sometimes refer to it and often use the old OSI layer numbers. The IETF has repeatedly stated that Internet protocol and architecture development is not intended to be OSI-compliant. RFC 3439, addressing Internet architecture, contains a section entitled: "Layering Considered Harmful".[18]

References

[1] Architectural Principles of the Internet (ftp://ftp.rfc-editor.org/in-notes/rfc1958.txt), RFC 1958, B. Carpenter, June 1996

[2] Rethinking the design of the Internet: The end to end arguments vs. the brave new world (http://www.csd.uoc.gr/~hy558/papers/Rethinking_2001.pdf), Marjory S. Blumenthal, David D. Clark, August 2001

[3] p.23 INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September 1981 Jon Postel Editor (http://www.ietf.org/rfc/rfc0791.txt?number=791)

[4] Requirements for Internet Hosts -- Communication Layers p.13 October 1989 R. Braden, Editor (http://tools.ietf.org/html/rfc1122#page-12)

[5] [OSI: Reference Model Addendum 1: Connectionless-mode Transmission,ISO7498/AD1],ISO7498/AD1, May 1986

[6] Internal Organization of the Network Layer (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16011), ISO 8648

[7] Open Systems Interconnection -- Basic Reference Model -- Part 4: Management framework (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=14258&ICS1=35&ICS2=100&ICS3=70), ISO 7498/4

[8] IP Packet Structure (http://www.comsci.us/datacom/ippacket.html)

[9] *TCP/IP Illustrated: the protocols* (http://www.kohala.com/start/tcpipiv1.html), ISBN 0-201-63346-9, W. Richard Stevens, February 1994

[10] Framework and taxonomy of International Standardized Profiles (http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=30726), ISO 10000, October 1998

[11] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach, 2008, ISBN 0-321-49770-8 (http://www.pearsonhighered.com/educator/academic/product/0,,0321497708,00+en-USS_01DBC.html)

[12] Behrouz A. Forouzan, Data Communications and Networking, 2003 (http://books.google.com/books?id=U3Gcf65Pu9IC&printsec=frontcover&dq=forouzan+"computer+networks"&ei=RPZ9SOCvMofctAO02di0AQ&hl=en&sig=ACfU3U2Hh_n83pPtf5uCreCih0HnWvNcxg#PPA29,M1)

[13] Douglas E. Comer, Internetworking with TCP/IP: Principles, Protocols and Architecture, Pearson Prentice Hall 2005, ISBN 0-13-187671-6 (http://books.google.com/books?id=jonyuTASbWAC&pg=PA155&hl=sv&source=gbs_toc_r&cad=0_0&sig=ACfU3U18gHAia1pU_Pxn-rhkCnH1v70M6Q#PPA161,M1)

[14] Charles M. Kozierok, "The TCP/IP Guide", No Starch Press 2005 (http://books.google.com/books?id=Pm4RgYV2w4YC&pg=PA131&dq="TCP/IP+model+layers"&lr=&hl=sv&sig=ACfU3U3ofMwYAbZfGz1BmAXc2oNNFC2b8A#PPA129,M1)

[15] William Stallings, Data and Computer Communications, Prentice Hall 2006, ISBN 0-13-243310-9 (http://books.google.com/books?id=c_AWmhkovR0C&pg=PA35&dq="internet+layer"+"network+access+layer"&ei=-O99SI3EJo32sgOQpPThDw&hl=en&sig=ACfU3U38aXznzeAnQdbLcPFXfCgxAd4lFg)

[16] Andrew S. Tanenbaum, Computer Networks, Prentice Hall 2002, ISBN 0-13-066102-3 (http://books.google.com/books?id=Pd-z64SJRBAC&pg=PA42&vq=internet+layer&dq=networks&hl=sv&source=gbs_search_s&sig=ACfU3U3DHANeIz0sOsd5NK4VXSrgNFYVAw#PPA42,M1)

[17] Mark Dye, Mark A. Dye, Wendell, Network Fundamentals: CCNA Exploration Companion Guide, 2007, ISBN 1-58713-208-7

[18] R. Bush; D. Meyer (December 2002), *Some Internet Architectural Guidelines and Philosophy* (http://www.isi.edu/in-notes/rfc3439.txt), Internet Engineering Task Force, , retrieved 2007-11-20

Further reading

- Clark, D. (1988). "The Design Philosophy of the DARPA Internet Protocols" (http://www.cs.princeton.edu/~jrex/teaching/spring2005/reading/clark88.pdf). *SIGCOMM '88 Symposium proceedings on Communications architectures and protocols* (ACM): 106–114. doi:10.1145/52324.52336. Retrieved 2011-10-16.

External links

- Understanding Cisco TCP/IP (http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ap1.htm)

Link Layer

Address Resolution Protocol

Address Resolution Protocol (**ARP**) is a telecommunications protocol used for resolution of network layer addresses into link layer addresses, a critical function in multiple-access networks. ARP was defined by RFC 826 in 1982.[1] It is Internet Standard STD 37. It is also the name of the program for manipulating these addresses in most operating systems.

ARP has been implemented in many combinations of network and overlaying internetwork technologies, such as IPv4, Chaosnet, DECnet and Xerox PARC Universal Packet (PUP) using IEEE 802 standards, FDDI, X.25, Frame Relay and Asynchronous Transfer Mode (ATM), IPv4 over IEEE 802.3 and IEEE 802.11 being the most common cases.

In Internet Protocol Version 6 (IPv6) networks, the functionality of ARP is provided by the Neighbor Discovery Protocol (NDP).

Operating scope

The Address Resolution Protocol is a request and reply protocol that runs encapsulated by the line protocol. It is communicated within the boundaries of a single network, never routed across internetwork nodes. This property places ARP into the Link Layer of the Internet Protocol Suite, while in the Open Systems Interconnect(OSI) model, it is often described as residing between Layers 2 and 3, being encapsulated by Layer 2 protocols. However, ARP was not developed in the OSI framework.

Packet structure

The Address Resolution Protocol uses a simple message format that contains one address resolution request or response. The size of the ARP message depends on the upper layer and lower layer address sizes, which are given by the type of networking protocol (usually IPv4) in use and the type of hardware or virtual link layer that the upper layer protocol is running on. The message header specifies these types, as well as the size of addresses of each. The message header is completed with the operation code for request (1) and reply (2). The payload of the packet consists of four addresses, the hardware and protocol address of the sender and receiver hosts.

The principal packet structure of ARP packets is shown in the following table which illustrates the case of IPv4 networks running on Ethernet. In this scenario, the packet has 48-bit fields for the sender hardware address (SHA) and target hardware address (THA), and 32-bit fields for the corresponding sender and target protocol addresses (SPA and TPA). Thus, the ARP packet size in this case is 28 bytes. The EtherType for ARP is 0x806.

Internet Protocol (IPv4) over Ethernet ARP packet		
bit offset	**0 – 7**	**8 – 15**
0	Hardware type (HTYPE)	
16	Protocol type (PTYPE)	
32	Hardware address length (HLEN)	Protocol address length (PLEN)
48	Operation (OPER)	
64	Sender hardware address (SHA) (first 16 bits)	
80	(next 16 bits)	
96	(last 16 bits)	
112	Sender protocol address (SPA) (first 16 bits)	
128	(last 16 bits)	
144	Target hardware address (THA) (first 16 bits)	
160	(next 16 bits)	
176	(last 16 bits)	
192	Target protocol address (TPA) (first 16 bits)	
208	(last 16 bits)	

Hardware type (HTYPE)

This field specifies the network protocol type. Example: Ethernet is 1.

Protocol type (PTYPE)

This field specifies the internetwork protocol for which the ARP request is intended. For IPv4, this has the value 0x0800. The permitted PTYPE values share a numbering space with those for EtherType.[2] [3] [4]

Hardware length (HLEN)

Length (in octets) of a hardware address. Ethernet addresses size is 6.

Protocol length (PLEN)

Length (in octets) of addresses used in the upper layer protocol. (The upper layer protocol specified in PTYPE.) IPv4 address size is 4.

Operation

Specifies the operation that the sender is performing: 1 for request, 2 for reply.

Sender hardware address (SHA)

media address of the sender.

Sender protocol address (SPA)

internetwork address of the sender.

Target hardware address (THA)

media address of the intended receiver. This field is ignored in requests.

Target protocol address (TPA)

internetwork address of the intended receiver.

ARP protocol parameter values have been standardized and are maintained by the Internet Assigned Numbers Authority (IANA).[5]

Example

For example, the computers *Matterhorn* and *Washington* are in an office, connected to each other on the office local area network by Ethernet cables and network switches, with no intervening gateways or routers. Matterhorn wants to send a packet to Washington. Through other means, it determines that Washington's IP address is 192.168.0.55. In order to send the message, it also needs to know Washington's MAC address. First, Matterhorn uses a cached ARP table to look up 192.168.0.55 for any existing records of Washington's MAC address (00:eb:24:b2:05:ac). If the MAC address is found, it sends the IP packet on the link layer to address 00:eb:24:b2:05:ac via the local network cabling. If the cache did not produce a result for 192.168.0.55, Matterhorn has to send a broadcast ARP message (destination FF:FF:FF:FF:FF:FF) requesting an answer for 192.168.0.55. Washington responds with its MAC address (00:eb:24:b2:05:ac). Washington may insert an entry for Matterhorn into its own ARP table for future use. The response information is cached in Matterhorn's ARP table and the message can now be sent.

ARP probe

An **ARP probe** is an ARP request constructed with an all-zero *sender IP address*. The term is used in the *IPv4 Address Conflict Detection* specification (RFC 5227). Before beginning to use an IPv4 address (whether received from manual configuration, DHCP, or some other means), a host implementing this specification must test to see if the address is already in use, by broadcasting ARP probe packets.

ARP announcements

ARP may also be used as a simple announcement protocol. This is useful for updating other hosts' mapping of a hardware address when the sender's IP address or MAC address has changed. Such an announcement, also called a *gratuitous ARP* message, is usually broadcast as an ARP request containing the sender's protocol address (SPA) in the target field (TPA=SPA), with the target hardware address (THA) set to zero. An alternative is to broadcast an ARP reply with the sender's hardware and protocol addresses (SHA and SPA) duplicated in the target fields (TPA=SPA, THA=SHA).

An ARP announcement is not intended to solicit a reply; instead it updates any cached entries in the ARP tables of other hosts that receive the packet. The operation code may indicate a request or a reply because the ARP standard specifies that the opcode is only processed after the ARP table has been updated from the address fields.[6] [7] [8]

Many operating systems perform gratuitous ARP during startup. That helps to resolve problems which would otherwise occur if, for example, a network card was recently changed (changing the IP-address-to-MAC-address mapping) and other hosts still have the old mapping in their ARP caches.

Gratuitous ARP is also used by some interface drivers to provide load balancing for incoming traffic. In a team of network cards, it is used to announce a different MAC address within the team that should receive incoming packets.

ARP announcements can be used to defend link-local IP addresses in the Zeroconf protocol (RFC 3927), and for IP address takeover within high-availability clusters.

ARP mediation

ARP mediation refers to the process of resolving Layer 2 addresses when different resolution protocols are used on multiple connected circuits, e.g., ATM on one end and Ethernet on the others. A proposed standard for ARP mediation is currently in draft status under the Internet Engineering Task Force.[9]

Inverse ARP and Reverse ARP

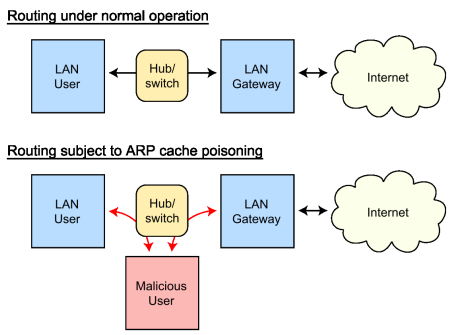
Inverse Address Resolution Protocol (Inverse ARP or InARP) is used to obtain Network Layer addresses (for example, IP addresses) of other nodes from Data Link Layer (Layer 2) addresses. It is primarily used in Frame Relay (DLCI) and ATM networks, in which Layer 2 addresses of virtual circuits are sometimes obtained from Layer 2 signaling, and the corresponding Layer 3 addresses must be available before those virtual circuits can be used.[10]

Since ARP translates Layer 3 addresses to Layer 2 addresses, InARP may be described as its inverse. In addition, InARP is implemented as a protocol extension to ARP: it uses the same packet format as ARP, but different operation codes.

The Reverse Address Resolution Protocol (Reverse ARP or RARP), like InARP, translates Layer 2 addresses to Layer 3 addresses. However, in InARP the requesting station queries the Layer 3 address of another node, whereas RARP is used to obtain the Layer 3 address of the requesting station itself for address configuration purposes. RARP is obsolete; it was replaced by BOOTP, which was later superseded by the Dynamic Host Configuration Protocol (DHCP).[11]

ARP spoofing and Proxy ARP

Because ARP does not provide methods for authenticating ARP replies on a network, ARP replies can come from systems other than the one with the required Layer 2 address. An ARP *proxy* is a system which answers the ARP request on behalf of another system for which it will forward traffic, normally as part of network design such as dialup internet service. By contrast in ARP *spoofing*, where the spoofer answers the ARP requests with the aim of interception. A malicious user may leverage ARP spoofing to perform a man-in-the-middle or denial-of-service attack on other users on the network. Various software exists to both detect and perform ARP spoofing attacks, though ARP itself does not provide any methods of protection from such attacks.[12]



A successful ARP spoofing attack allows an attacker to perform a man-in-the-middle attack.

Alternatives to ARP

Each computer maintains its own table of the mapping from Layer 3 addresses (e.g. IP addresses) to Layer 2 addresses (e.g. ethernet MAC addresses). In a modern computer this is maintained almost entirely by ARP packets on the local network and it thus often called the 'ARP cache' as opposed to 'Layer 2 address table'. In older computers, where broadcast packets were considered an expensive resource, other methods were used to maintain this table, such as static configuration files[13] or centrally maintained lists. Since at least the 1980s[14] , networked computers have had a command called *arp* for interrogating or manipulating this table, and practically all modern personal computers have a variant of this[15] [16] [17] [18] .

ARP Stuffing

Embedded systems such as networked cameras[19] and networked power distribution devices[20] , which lack a user interface, can use so-called *ARP stuffing* to make an initial network connection, although this is a misnomer as there is no ARP *protocol* involved. This is a solution to an issue in network management of consumer devices, specifically the allocation of IP addresses of ethernet devices where 1) the user doesn't have the ability to control DHCP or similar address allocation protocols, 2) the device doesn't have a user interface to configure it, and 3) the user's computer can't communicate with it because it has no suitable IP address.

The solution adopted is as follows: the user's computer has an IP address *stuffed* manually into its address table (normally with the *arp* command with the MAC address taken from a label on the device) and then sends special packets to the device, typically a ping packet with a non-default size. The device then adopts this IP address, and the user then communicates with it by telnet or web protocols to complete the configuration. Such devices typically have a method to disable this process once the device is operating normally, as it is open to Denial of Service attack.

References

[1] David C. Plummer (1982-11). "RFC 826, An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware" (http://tools.ietf.org/html/rfc826). Internet Engineering Task Force, Network Working Group. .

[2] IANA ARP - "Protocol Type" (http://www.iana.org/assignments/arp-parameters/)

[3] IANA - Ethertype values (http://www.iana.org/assignments/ethernet-numbers)

[4] RFC 5342

[5] "IANA ARP parameter assignments" (http://www.iana.org/assignments/arp-parameters/). IANA. 2009-04-24. .

[6] Gratuitous ARP in DHCP vs. IPv4 ACD Draft (http://www1.ietf.org/mail-archive/web/dhcwg/current/msg03797.html)

[7] RFC 2002 Section 4.6 (http://tools.ietf.org/html/rfc2002#section-4.6)

[8] RFC 2131 DHCP – Last lines of Section 4.4.1 (http://tools.ietf.org/html/rfc2131#section-4.4.1)

[9] Himanshu Shah, et. al. (2011-04-03). "ARP Mediation for IP Interworking of Layer 2 VPN" (http://tools.ietf.org/html/draft-ietf-l2vpn-arp-mediation-16). Internet Engineering Task Force. .

[10] T. Bradley, et. al. (1998-09). "RFC 2390 - Inverse Address Resolution Protocol" (http://tools.ietf.org/html/rfc2390). Internet Engineering Task Force. .

[11] Finlayson, Mann, Mogul, Theimer (1984-06). "RFC 903 - A Reverse Address Resolution Protocol" (http://tools.ietf.org/html/rfc903). Internet Engineering Task Force. .

[12] Steve Gibson (2005-12-11). "ARP Cache Poisoning" (http://www.grc.com/nat/arp.htm). GRC. .

[13] Sun Microsystems. "SunOS manual page for ethers(5) file" (http://www.freebsd.org/cgi/man.cgi?query=ethers&sektion=5&apropos=0&manpath=SunOS+4.1.3). . Retrieved 2011-09-28.

[14] University of California, Berkeley. "BSD manual page for arp(8C) command" (http://www.freebsd.org/cgi/man.cgi?query=arp&apropos=0&sektion=0&manpath=2.10+BSD&arch=default&format=html). . Retrieved 2011-09-28.

[15] Canonical. "Ubuntu manual page for arp(8) command" (http://manpages.ubuntu.com/manpages/lucid/man8/arp.8.html). . Retrieved 2011-09-28.

[16] Apple Computer. "Mac OSX manual page for arp(8) command" (http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man8/arp.8.html). . Retrieved 2011-09-28.

[17] Microsoft. "Windows help for arp command" (http://technet.microsoft.com/en-us/library/cc786759(WS.10).aspx). . Retrieved 2011-09-28.

[18] Cisco. "Cisco reference for 'show ip arp' command" (http://www.cisco.com/en/US/docs/ios/12_3/ipaddr/command/reference/ip1_s1g.html#wp1079902). . Retrieved 2011-09-28.

[19] Axis Communication. "Axis P13 Network Camera Series Installation Guide" (http://www.axis.com/files/manuals/ig_p13Series_38731_en_1006.pdf). . Retrieved 2011-09-28.

[20] American Power Corporation. "Switched Rack Power Distribution Unit Installation and Quick Start Manual" (http://www.apcmedia.com/salestools/ASTE-6Z6K56_R0_EN.pdf). . Retrieved 2011-09-28.

This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.

External links

- RFC 826 - Ethernet Address Resolution Protocol, Internet Standard STD 37.
- RFC 903 - Reverse Address Resolution Protocol, Internet Standard STD 38.
- RFC 2390 - Inverse Address Resolution Protocol, draft standard
- RFC 5227 - IPv4 Address Conflict Detection, proposed standard
- ArpON home page (http://arpon.sourceforge.net)
- ARP Sequence Diagram (pdf) (http://www.eventhelix.com/RealtimeMantra/Networking/Arp.pdf)
- Gratuitous ARP (http://wiki.wireshark.org/Gratuitous_ARP)
- Free ARP tools with source code (French) (http://www.authsecu.com/arpflood/)
- ARP-SK ARP traffic generation tools (http://sid.rstack.org/arp-sk/)
- Sample Capture file from WireSharkWiki (http://wiki.wireshark.org/SampleCaptures#head-2fb4a82886c1d8c722134b44461e22e5f7f54b32)

Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a standard switching technique designed to unify telecommunication and computer networks. It uses asynchronous time-division multiplexing,[1] [2] and it encodes data into small, fixed-sized *cells.* This differs from approaches such as the Internet Protocol or Ethernet that use variable sized *packets* or *frames.* ATM provides data link layer services that run over a wide range of OSI physical Layer links. ATM has functional similarity with both circuit switched networking and small packet switched networking. It was designed for a network that must handle both traditional high-throughput data traffic (e.g., file transfers), and real-time, low-latency content such as voice and video. ATM uses a connection-oriented model in which a virtual circuit must be established between two endpoints before the actual data exchange begins.[2] ATM is a core protocol used over the SONET/SDH backbone of the public switched telephone network (PSTN) and Integrated Services Digital Network (ISDN), but its use is declining in favour of All IP.

ATM concepts

ATM was developed to meet the needs of the Broadband Integrated Services Digital Network, as defined in the late 1980s.[3]

Why cells?

IBM Turboways ATM 155 PCI network interface card

Consider a speech signal reduced to packets, and forced to share a link with bursty data traffic (traffic with some large data packets). No matter how small the speech packets could be made, they would always encounter full-size data packets, and under normal queuing conditions, might experience maximum queuing delays. That is why all packets, or "cells," should have the same small size. In addition the fixed cell structure means that ATM can be readily switched by hardware without the inherent delays introduced by software switched and routed frames.

Thus, the designers of ATM utilized small data cells to reduce jitter (delay variance, in this case) in the multiplexing of data streams. Reduction of jitter (and also end-to-end round-trip delays) is particularly important when carrying voice traffic, because the conversion of digitized voice into an analogue audio signal is an inherently real-time process, and to do a good job, the decoder (codec) that does this needs an evenly spaced (in time) stream of data items. If the next data item is not available when it is needed, the codec has no choice but to produce silence or guess — and if the data is late, it is useless, because the time period when it should have been converted to a signal has already passed.

At the time of the design of ATM, 155 Mbit/s Synchronous Digital Hierarchy (SDH) with 135 Mbit/s payload was considered a fast optical network link, and many Plesiochronous Digital Hierarchy (PDH) links in the digital network were considerably slower, ranging from 1.544 to 45 Mbit/s in the USA, and 2 to 34 Mbit/s in Europe.

At this rate, a typical full-length 1500 byte (12000-bit) data packet would take 77.42 µs to transmit. In a lower-speed link, such as a 1.544 Mbit/s T1 line, a 1500 byte packet would take up to 7.8 milliseconds.

A queuing delay induced by several such data packets might exceed the figure of 7.8 ms several times over, in addition to any packet generation delay in the shorter speech packet. This was clearly unacceptable for speech traffic, which needs to have low jitter in the data stream being fed into the codec if it is to produce good-quality sound. A packet voice system can produce this in a number of ways:

- Have a playback buffer between the network and the codec, one large enough to tide the codec over almost all the jitter in the data. This allows smoothing out the jitter, but the delay introduced by passage through the buffer would require echo cancellers even in local networks; this was considered too expensive at the time. Also, it would have increased the delay across the channel, and conversation is difficult over high-delay channels.
- Build a system which can inherently provide low jitter (and minimal overall delay) to traffic which needs it.
- Operate on a 1:1 user basis (i.e., a dedicated pipe).

The design of ATM aimed for a low-jitter network interface. However, "cells" were introduced into the design to provide short queuing delays while continuing to support datagram traffic. ATM broke up all packets, data, and voice streams into 48-byte chunks, adding a 5-byte routing header to each one so that they could be reassembled later. The choice of 48 bytes was political rather than technical.[4] When the CCITT (now ITU-T) was standardizing ATM, parties from the United States wanted a 64-byte payload because this was felt to be a good compromise in larger payloads optimized for data transmission and shorter payloads optimized for real-time applications like voice; parties from Europe wanted 32-byte payloads because the small size (and therefore short transmission times) simplify voice applications with respect to echo cancellation. Most of the European parties eventually came around to the arguments made by the Americans, but France and a few others held out for a shorter cell length. With 32 bytes, France would have been able to implement an ATM-based voice network with calls from one end of France to the other requiring no echo cancellation. 48 bytes (plus 5 header bytes = 53) was chosen as a compromise between the two sides. 5-byte headers were chosen because it was thought that 10% of the payload was the maximum price to pay for routing information.[3] ATM multiplexed these 53-byte cells instead of packets which reduced worst-case cell contention jitter by a factor of almost 30, reducing the need for echo cancellers.

Cells in practice

ATM supports different types of services via ATM adaptation layers (AAL). Standardized AALs include AAL1, AAL2, and AAL5, and the rarely used AAL3 and AAL4. AAL1 is used for constant bit rate (CBR) services and circuit emulation. Synchronization is also maintained at AAL1. AAL2 through AAL4 are used for variable bit rate (VBR) services, and AAL5 for data. Which AAL is in use for a given cell is not encoded in the cell. Instead, it is negotiated by or configured at the endpoints on a per-virtual-connection basis.

Following the initial design of ATM, networks have become much faster. A 1500 byte (12000-bit) full-size Ethernet frame takes only 1.2 µs to transmit on a 10 Gbit/s network, reducing the need for small cells to reduce jitter due to

contention. Some consider that this makes a case for replacing ATM with Ethernet in the network backbone. However, it should be noted that the increased link speeds by themselves do not alleviate jitter due to queuing. Additionally, the hardware for implementing the service adaptation for IP packets is expensive at very high speeds. Specifically, at speeds of OC-3 and above, the cost of segmentation and reassembly (SAR) hardware makes ATM less competitive for IP than Packet Over SONET (POS); because of its fixed 48-byte cell payload, ATM is not suitable as a data link layer *directly* underlying IP (without the need for SAR at the data link level) since the OSI layer on which IP operates must provide an maximum transmission unit (|MTU) of at least 576 bytes. SAR performance limits mean that the fastest IP router ATM interfaces are STM16 - STM64 which actually compares, while as of 2004 POS can operate at OC-192 (STM64) with higher speeds expected in the future.

On slower or congested links (622 Mbit/s and below), ATM does make sense, and for this reason most asymmetric digital subscriber line (ADSL) systems use ATM as an intermediate layer between the physical link layer and a Layer 2 protocol like PPP or Ethernet.

At these lower speeds, ATM provides a useful ability to carry multiple logical circuits on a single physical or virtual medium, although other techniques exist, such as Multi-link PPP and Ethernet VLANs, which are optional in VDSL implementations. DSL can be used as an access method for an ATM network, allowing a DSL termination point in a telephone central office to connect to many internet service providers across a wide-area ATM network. In the United States, at least, this has allowed DSL providers to provide DSL access to the customers of many internet service providers. Since one DSL termination point can support multiple ISPs, the economic feasibility of DSL is substantially improved.

Why virtual circuits?

ATM operates as a channel-based transport layer, using virtual circuits (VCs). This is encompassed in the concept of the Virtual Paths (VP) and Virtual Channels. Every ATM cell has an 8- or 12-bit **Virtual Path Identifier** (VPI) and 16-bit Virtual Channel Identifier (VCI) pair defined in its header.[5] Together, these identify the virtual circuit used by the connection. The length of the VPI varies according to whether the cell is sent on the user-network interface (on the edge of the network), or if it is sent on the network-network interface (inside the network).

As these cells traverse an ATM network, switching takes place by changing the VPI/VCI values (label swapping). Although the VPI/VCI values are not necessarily consistent from one end of the connection to the other, the concept of a circuit *is* consistent (unlike IP, where any given packet could get to its destination by a different route than the others).[6]

Another advantage of the use of virtual circuits comes with the ability to use them as a multiplexing layer, allowing different services (such as voice, Frame Relay, n* 64 channels, IP). The VPI is useful for reducing the switching table of some virtual circuits which have common paths.

Using cells and virtual circuits for traffic engineering

Another key ATM concept involves the traffic contract. When an ATM circuit is set up each switch on the circuit is informed of the traffic class of the connection.

ATM traffic contracts form part of the mechanism by which "quality of service" (QoS) is ensured. There are four basic types (and several variants) which each have a set of parameters describing the connection.

1. CBR - Constant bit rate: a Peak Cell Rate (PCR) is specified, which is constant.
2. VBR - Variable bit rate: an average cell rate is specified, which can peak at a certain level for a maximum interval before being problematic.
3. ABR - Available bit rate: a minimum guaranteed rate is specified.
4. UBR - Unspecified bit rate: traffic is allocated to all remaining transmission capacity.

VBR has real-time and non-real-time variants, and serves for "bursty" traffic. Non-real-time is sometimes abbreviated to vbr-nrt.

Most traffic classes also introduce the concept of Cell Delay Variation Tolerance (CDVT), which defines the "clumping" of cells in time.

To maintain traffic contracts, networks usually use "shaping", a combination of queuing and marking of cells. "Traffic policing" generally enforces traffic contracts.

Traffic shaping

Traffic shaping usually takes place at the entry point to an ATM network and attempts to ensure that the cell flow will meet its traffic contract.

Traffic policing

To maintain network performance, networks may police virtual circuits against their traffic contracts. If a circuit is exceeding its traffic contract, the network can either drop the cells or mark the Cell Loss Priority (CLP) bit (to identify a cell as potentially redundant). Basic policing works on a cell by cell basis, but this is sub-optimal for encapsulated packet traffic (as discarding a single cell will invalidate the whole packet). As a result, schemes such as Partial Packet Discard (PPD) and Early Packet Discard (EPD) have been created that will discard a whole series of cells until the next frame starts. This reduces the number of useless cells in the network, saving bandwidth for full frames. EPD and PPD work with AAL5 connections as they use the frame end bit to detect the end of packets.

Types of virtual circuits and paths

ATM can build virtual circuits and virtual paths either statically or dynamically. Static circuits (permanent virtual circuits or PVCs) or paths (permanent virtual paths or PVPs) require that the circuit is provided as a series of segments, one for each pair of interfaces through which it passes.

PVPs and PVCs, though conceptually simple, require significant effort in large networks. They also do not support the re-routing of service in the event of a failure. Dynamically built PVPs (soft PVPs or SPVPs) and PVCs (soft PVCs or SPVCs), in contrast, are built by specifying the characteristics of the circuit (the service "contract") and the two end points.

Finally, ATM networks create and remove switched virtual circuits (SVCs) on demand when requested by an end piece of equipment. One application for SVCs is to carry individual telephone calls when a network of telephone switches are inter-connected using ATM. SVCs were also used in attempts to replace local area networks with ATM.

Virtual circuit routing

Most ATM networks supporting SPVPs, SPVCs, and SVCs use the Private Network Node Interface or Private Network-to-Network Interface (PNNI) protocol. PNNI uses the same shortest-path-first algorithm used by OSPF and IS-IS to route IP packets to share topology information between switches and select a route through a network. PNNI also includes a very powerful summarization mechanism to allow construction of very large networks, as well as a call admission control (CAC) algorithm which determines the availability of sufficient bandwidth on a proposed route through a network in order to satisfy the service requirements of a VC or VP.

Call admission and connection establishment

A network must establish a connection before two parties can send cells to each other. In ATM this is called a virtual circuit (VC). It can be a permanent virtual circuit (PVC), which is created administratively on the end points, or a switched virtual circuit (SVC), which is created as needed by the communicating parties. SVC creation is managed by signaling, in which the requesting party indicates the address of the receiving party, the type of service requested, and whatever traffic parameters may be applicable to the selected service. "Call admission" is then performed by the network to confirm that the requested resources are available and that a route exists for the connection.

Reference model

ATM defines three layers:[7]

1. **ATM Adaptation Layer (AAL)** and
2. **ATM Layer** correspond roughly to the OSI Data Link Layer
3. **Physical Layer**, equivalent to OSI Physical Layer

Structure of an ATM cell

An ATM cell consists of a 5-byte header and a 48-byte payload. The payload size of 48 bytes was chosen as described above ("Why cells?").

ATM defines two different cell formats: UNI (User-Network Interface) and NNI (Network-Network Interface). Most ATM links use UNI cell format.

Diagram of the UNI ATM Cell

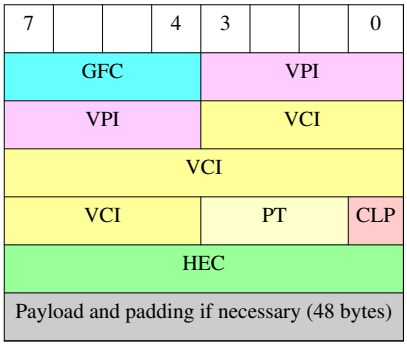
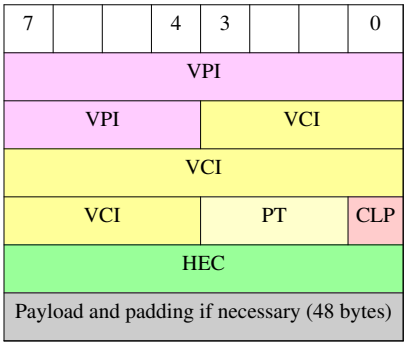


Diagram of the NNI ATM Cell



GFC = Generic Flow Control (4 bits) (default: 4-zero bits)

VPI = Virtual Path Identifier (8 bits UNI) or (12 bits NNI)

VCI = Virtual Channel identifier (16 bits)

PT = Payload Type (3 bits)

CLP = Cell Loss Priority (1-bit)

HEC = Header Error Control (8-bit CRC, polynomial = $X^8 + X^2 + X + 1$)

ATM uses the PT field to designate various special kinds of cells for operations, administration and management (OAM) purposes, and to delineate packet boundaries in some AALs.

Several ATM link protocols use the HEC field to drive a CRC-based framing algorithm, which allows locating the ATM cells with no overhead beyond what is otherwise needed for header protection. The 8-bit CRC is used to correct single-bit header errors and detect multi-bit header errors. When multi-bit header errors are detected, the current and subsequent cells are dropped until a cell with no header errors is found.

A UNI cell reserves the GFC field for a local flow control/submultiplexing system between users. This was intended to allow several terminals to share a single network connection, in the same way that two Integrated Services Digital Network (ISDN) phones can share a single basic rate ISDN connection. All four GFC bits must be zero by default.

The NNI cell format replicates the UNI format almost exactly, except that the 4-bit GFC field is re-allocated to the VPI field, extending the VPI to 12 bits. Thus, a single NNI ATM interconnection is capable of addressing almost 2^{12} VPs of up to almost 2^{16} VCs each (in practice some of the VP and VC numbers are reserved).

Deployment

ATM became popular with telephone companies and many computer makers in the 1990s. However, even by the end of the decade, the better price/performance of Internet Protocol-based products was competing with ATM technology for integrating real-time and bursty network traffic.[8] Companies such as FORE Systems focused on ATM products, while other large vendors such as Cisco Systems provided ATM as an option.[9] After the burst of the dot-com bubble, some still predicted that "ATM is going to dominate".[10] However, in 2005 the ATM Forum, which had been the trade organization promoting the technology, merged with groups promoting other technologies, and eventually became the Broadband Forum.[11]

Notes

[1] "Recommendation I.150, B-ISDN asynchronous transfer mode functional characteristics" (http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-I.150-199902-I!!PDF-E&type=items). ITU. .

[2] McDysan (1999), p. 287

[3] Ayanoglu, Ender; Akar, Nail. "B-ISDN (Broadband Integrated Services Digital Network)" (http://repositories.cdlib.org/cpcc/2/). Center for Pervasive Communications and Computing, UC Irvine. . Retrieved 3 June 2011.

[4] D. Stevenson, "Electropolitical Correctness and High-Speed Networking, or, Why ATM is like a Nose", Proceedings of TriCom '93 (http://www.amazon.co.uk/dp/0306444860), April 1993.

[5] Cisco Systems *Guide to ATM Technology* (http://www.cisco.com/univercd/cc/td/doc/product/atm/c8540/12_1/pereg_1/atm_tech/techgd.pdf) (2000). Section "Operation of an ATM Switch". Retrieved 2011-06-02.

[6] Cisco Systems *Guide to ATM Technology* (http://www.cisco.com/univercd/cc/td/doc/product/atm/c8540/12_1/pereg_1/atm_tech/techgd.pdf) (2000). Section "ATM Cell Header Formats". Retrieved 2011-06-02.

[7] "Guide to ATM Technology for the Catalyst 8540 MSR, Catalyst 8510 MSR, and LightStream 1010 ATM Switch Routers" (http://www.cisco.com/univercd/cc/td/doc/product/atm/c8540/12_1/pereg_1/atm_tech/techgd.pdf). *Customer Order Number: DOC-786275*. Cisco Systems. 2000. . Retrieved 19 July 2011.

[8] Steve Steinberg (October 1996). "Netheads vs Bellheads" (http://www.wired.com/wired/archive/4.10/atm.html). *Wired* **4** (10). . Retrieved 24 September 2011.

[9] "What's in store for FORE?" (http://books.google.com/books?id=IxgEAAAAMBAJ&lpg=PA12). *Network World*: p. 12. 16 September 1996. . Retrieved 24 September 2011.

[10] "Optical Ethernet firms brave stormy industry seas" (http://books.google.com/books?id=GBwEAAAAMBAJ&pg=PA14). *Network World*: p. 14. 7 May 2001. . Retrieved 24 September 2011.

[11] "About the Broadband Forum: Forum History" (http://www.broadband-forum.org/about/forumhistory.php). . Retrieved 24 September 2011.

References

- Black, Uyless D. (1998). *ATM—Volume III: Internetworking with ATM*. Toronto: Prentice Hall. ISBN 0-13-784182-5.
- De Prycker, Martin (1993). *Asynchronous Transfer Mode. Solutions for Broadband ISDN*. Prentice Hall.
- Joel, Amos E., Jr. (1993). *Asynchronous Transfer Mode*. IEEE Press.
- Golway, Tom (1997). *Planning and Managing ATM Networks*. New York: Manning. ISBN 9780132621892.
- McDysan, David E.; Darren L. Spohn (1999). *ATM Theory and Applications*. Montreal: McGraw-Hill. ISBN 0-07-045346-2.
- Neelakanta, P. S. (2000). *A Textbook on ATM Telecommunications, Principles and implementation*. CRC Press. ISBN 0-8493-1805-X.

- ATM Cell formats- Cisco Systems (http://www.cisco.com/univercd/cc/td/doc/product/atm/c8540/12_0/13_19/trouble/cells.htm)
- Asynchronous Transfer Mode (ATM) - Cisco Systems (http://www.cisco.com/web/psa/technologies/index.html?c=268435599)

External links

- "ATM forum" (http://replay.web.archive.org/20050701081559/http://www.atmforum.com/). Archived from the original (http://www.atmforum.com/) on 2005-07-01.
- ATM Info and resources (http://www.telecomspace.com/vop-atm.html)
- ATM ChipWeb - Chip and NIC database (http://www.chipweb.de/atm/)
- *A tutorial from Juniper web site* (http://www.juniper.net/techpubs/software/erx/junose61/swconfig-link/html/atm-config2.html)
- ATM Tutorial (http://www2.rad.com/networks/2004/atm/main.htm)

Bootstrap Protocol

In computer networking, the **Bootstrap Protocol**, or **BOOTP**, is a network protocol used by a network client to obtain an IP address from a configuration server. The BOOTP protocol was originally defined in RFC 951.

BOOTP is usually used during the bootstrap process when a computer is starting up. A BOOTP configuration server assigns an IP address to each client from a pool of addresses. BOOTP uses the User Datagram Protocol (UDP) as a transport on IPv4 networks only.

Historically, BOOTP has also been used for Unix-like diskless workstations to obtain the network location of their boot image in addition to an IP address, and also by enterprises to roll out a pre-configured client (e.g., Windows) installation to newly installed PCs.

Originally requiring the use of a boot floppy disk to establish the initial network connection, manufacturers of network cards later embedded the protocol in the BIOS of the interface cards as well as system boards with on-board network adapters, thus allowing direct network booting.

Recently, users with an interest in diskless stand-alone media center PCs have shown new interest in this method of booting a Windows operating system.[1]

The Dynamic Host Configuration Protocol (DHCP) is a more advanced protocol for the same purpose and has superseded the use of BOOTP. Most DHCP servers also function as BOOTP servers.

History

The BOOTP protocol was first defined in RFC 951 as a replacement for the Reverse Address Resolution Protocol RARP, published in RFC 903 in June 1984. The primary motivation for replacing RARP with BOOTP is that RARP was a data link layer protocol. This made implementation difficult on many server platforms, and required that a server be present on each individual IP subnet. BOOTP introduced the innovation of a **relay agent**, which allowed BOOTP packets to be forwarded from the local network using standard IP routing, so that one central BOOTP server could serve hosts on many subnets.[2]

Related RFCs

BOOTP Related RFCs

Note that grayed out RFCs are obsolete

RFC #	Title	Date	Obsolete and Update Information
RFC 3942	Reclassifying Dynamic Host Configuration Protocol version 4 (DHCPv4) Options	Nov-04	Updates RFC 2132
RFC 2132	DHCP Options and BOOTP Vendor Extensions	Mar-97	Obsoletes RFC 1533, Updated by RFC 3442, RFC 3942, RFC 4361, RFC 4833, RFC 5494
RFC 1542	Clarifications and Extensions for the Bootstrap Protocol	Oct-93	Obsoletes RFC 1532, Updates RFC 951
RFC 1534	Interoperation Between DHCP and BOOTP	Oct-93	
RFC 1533	DHCP Options and BOOTP Vendor Extensions	Oct-93	Obsoletes RFC 1497, RFC 1395, RFC 1084, RFC 1048, Obsoleted by RFC 2132
RFC 1532	Clarifications and Extensions for the Bootstrap Protocol	Oct-93	Obsoleted by RFC 1542, Updates RFC 951
RFC 1497	BOOTP Vendor Information Extensions	Aug-93	Obsoletes RFC 1395, RFC 1084, RFC 1048, Obsoleted by RFC 1533, Updates RFC 951
RFC 1395	BOOTP Vendor Information Extensions	Jan-93	Obsoletes RFC 1084, RFC 1048, Obsoleted by RFC 1497, RFC 1533, Updates RFC 951
RFC 1084	BOOTP vendor information extensions	Dec-88	Obsoletes RFC 1048, Obsoleted by RFC 1395, RFC 1497, RFC 1533
RFC 1048	BOOTP vendor information extensions	Feb-88	Obsoleted by RFC 1084, RFC 1395, RFC 1497, RFC 1533
RFC 951	Bootstrap Protocol	Sep-85	Updated by RFC 1395, RFC 1497, RFC 1532, RFC 1542, RFC 5494

References

[1] Personal Computer World, Feb 2005, pg 156 'Putting the Boot in'

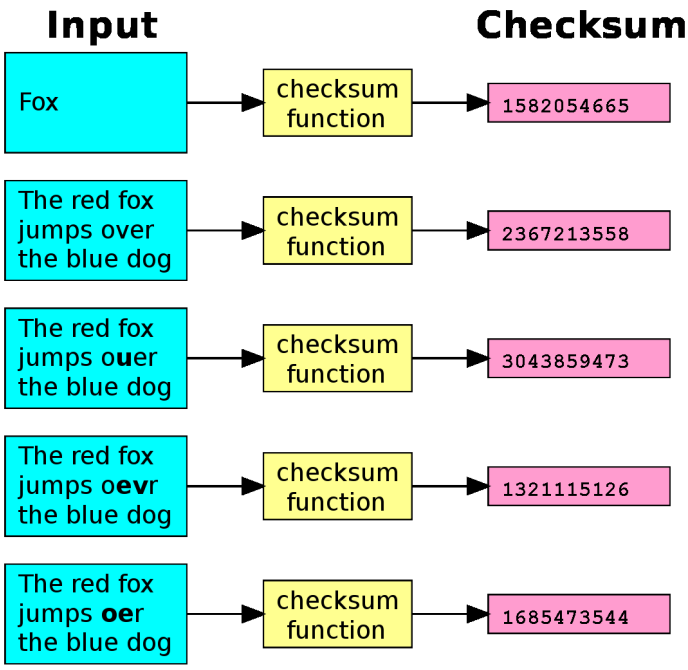
[2] Bill Croft; John Gilmore (September 1985). "RFC 951 - Bootstrap Protocol" (http://tools.ietf.org/html/rfc951#section-6). *Network Working Group*. .

External links

- BOOTP Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/Bootp.pdf) (PDF)
- Multicast BOOTP for configuring a network device from a workstation (http://mbootp.sourceforge.net/)

Checksum

A **checksum** or **hash sum** is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting accidental errors that may have been introduced during its transmission or storage. The integrity of the data can be checked at any later time by recomputing the checksum and comparing it with the stored one. If the checksums match, the data were almost certainly not altered (either intentionally or unintentionally).



Effect of a typical checksum function (the Unix `cksum` utility).

The procedure that yields the checksum from the data is called a **checksum function** or **checksum algorithm**. A good checksum algorithm will yield a different result with high probability when the data is accidentally corrupted; if the checksums match, the data is very likely to be free of accidental errors.

Checksum functions are related to hash functions, fingerprints, randomization functions, and cryptographic hash functions. However, each of those concepts has different applications and therefore different design goals. Check digits and parity bits are special cases of checksums, appropriate for small blocks of data (such as Social Security numbers, bank account numbers, computer words, single bytes, etc.). Some error-correcting codes are based on special checksums that not only detect common errors but also allow the original data to be recovered in certain cases.

Checksum algorithms

Parity byte or parity word

The simplest checksum algorithm is the so-called longitudinal parity check, which breaks the data into "words" with a fixed number n of bits, and then computes the exclusive or of all those words. The result is appended to the message as an extra word. To check the integrity of a message, the receiver computes the exclusive or of all its words, including the checksum; if the result is not a word with n zeros, the receiver knows that a transmission error occurred.

With this checksum, any transmission error that flips a single bit of the message, or an odd number of bits, will be detected as an incorrect checksum. However, an error that affects two bits will not be detected if those bits lie at the same position in two distinct words. If the affected bits are independently chosen at random, the probability of a two-bit error being undetected is $1/n$.

Modular sum

A variant of the previous algorithm is to add all the "words" as unsigned binary numbers, discarding any overflow bits, and append the two's complement of the total as the checksum. To validate a message, the receiver adds all the words in the same manner, including the checksum; if the result is not a word full of zeros, an error must have occurred. This variant too detects any single-bit error, but the probability that a two-bit error will go undetected is a little less than $1/n$.

Position-dependent checksums

The simple checksums described above fail to detect some common errors that affect many bits at once, such as changing the order of data words, or inserting or deleting words with all bits set to zero. The checksum algorithms that are most used in practice, such as Fletcher's checksum, Adler-32, and cyclic redundancy checks (CRCs), address these weaknesses by considering not only the value of each word but also its position in the sequence. This feature generally increases the cost of computing the checksum.

General considerations

A message that is m bits long can be viewed as a corner of the m-dimensional hypercube. The effect of a checksum algorithm that yields an n-bit checksum is to map each m-bit message to a corner of a larger hypercube, with dimension $m+n$. The 2^{m+n} corners of this hypercube represent all possible received messages. The valid received messages (those that have the correct checksum) comprise a smaller set, with only 2^m corners.

A single-bit transmission error then corresponds to a displacement from a valid corner (the correct message and checksum) to one of the m adjacent corners. An error that affects k bits moves the message to a corner that is k steps removed from its correct corner. The goal of a good checksum algorithm is to spread the valid corners as far from each other as possible, so as to increase the likelihood that "typical" transmission errors will end up in an invalid corner.

Checksum tools

- `cksum`, a Unix command that generates both a 32-bit CRC and a byte count for any given input file.
- `Bitser`, a free Microsoft Windows application that calculates MD5, SHA-1 and SHA-256 sums for any given input file.
- `md5sum`, a Unix command that generates an MD5 sum
- `jdigest` [1], a Java GUI tool that generates and checks MD5 and SHA sums
- `Jacksum` [2], a Java API, usable both through a GUI and a CLI, which incorporates many checksum implementations and allows to extend with as many as you need.
- `jcksum` [3], a Java library, that can be used by developers in Java applications to calculate checksums using different algorithms.
- `checksum` [4], "a blisteringly fast, no-nonsense file hashing application for Windows"

External links

- Additive Checksums (C) [5]

References

[1] http://code.google.com/p/jdigest/
[2] http://www.jonelo.de/java/jacksum/index.html
[3] http://code.google.com/p/jcksum/
[4] http://corz.org/windows/software/checksum
[5] http://www.netrino.com/Embedded-Systems/How-To/Additive-Checksums

Circuit switching

Circuit switching is a methodology of implementing a telecommunications network in which two network nodes establish a dedicated communications channel (circuit) through the network before the nodes may communicate. The circuit guarantees the full bandwidth of the channel and remains connected for the duration of the communication session. The circuit functions as if the nodes were physically connected as with an electrical circuit.

Circuit switching contrasts with packet switching which divides the data to be transmitted into packets transmitted through the network independently. Packet switching shares available network bandwidth between multiple communication sessions.

In circuit switching, the bit delay is constant during a connection, as opposed to packet switching, where packet queues may cause varying packet transfer delay. Each circuit cannot be used by other callers until the circuit is released and a new connection is set up. Even if no actual communication is taking place, the channel remains unavailable to other users. Channels that are available for new calls are said to be idle.

Virtual circuit switching is a packet switching technology that emulates circuit switching, in the sense that the connection is established before any packets are transferred, and packets are delivered in order.

While circuit switching is commonly used for connecting voice circuits, the concept of a dedicated path persisting between two communicating parties or nodes can be extended to signal content other than voice. Its advantage is that it provides for continuous transfer without the overhead associated with packets making maximal use of available bandwidth for that communication. The disadvantage is inflexibility; the connection and the bandwidth associated with it are reserved and unavailable for other uses.

The call

For call setup and control (and other administrative purposes), it is possible to use a separate dedicated signalling channel from the end node to the network. ISDN is one such service that uses a separate signalling channel while plain old telephone service (POTS) does not.

The method of establishing the connection and monitoring its progress and termination through the network may also utilize a separate control channel as in the case of links between telephone exchanges which use CCS7 packet-switched signalling protocol to communicate the call setup and control information and use TDM to transport the actual circuit data.

Early telephone exchanges are a suitable example of circuit switching. The subscriber would ask the operator to connect to another subscriber, whether on the same exchange or via an inter-exchange link and another operator. In any case, the end result was a physical electrical connection between the two subscribers' telephones for the duration of the call. The copper wire used for the connection could not be used to carry other calls at the same time, even if the subscribers were in fact not talking and the line was silent.

Compared to datagram packet switching

Circuit switching contrasts with packet switching which divides the data to be transmitted into small units, called packets, transmitted through the network independently. Packet switching shares available network bandwidth between multiple communication sessions.

Multiplexing multiple telecommunications connections over the same physical conductor has been possible for a long time, but nonetheless each channel on the multiplexed link was either dedicated to one call at a time, or it was idle between calls.

In circuit switching, and virtual circuit switching, a route and bandwidth is reserved from source to destination. Circuit switching can be relatively inefficient because capacity is guaranteed on connections which are set up but are not in continuous use, but rather momentarily. However, the connection is immediately available while established.

Packet switching is the process of segmenting a message/data to be transmitted into several smaller packets. Each packet is labeled with its destination and a sequence number for ordering related packets, precluding the need for a dedicated path to help the packet find its way to its destination. Each packet is dispatched independently and each may be routed via a different path. At the destination, the original message is reassembled in the correct order, based on the packet number. Datagram packet switching networks do not require a circuit to be established and allow many pairs of nodes to communicate concurrently over the same channel.

Examples of circuit-switched networks

- public switched telephone network (PSTN)
- ISDN B-channel
- Circuit Switched Data (CSD) and High-Speed Circuit-Switched Data (HSCSD) service in cellular systems such as GSM
- Datakit
- X.21 (Used in the German DATEX-L and Scandinavian DATEX circuit switched data network)

External links

- *Netheads vs Bellheads* by Steve Steinberg [1]
- University of Virginia [2]
- RFC 3489 *Some Internet Architectural Guidelines and Philopsophy* [3]

References

[1] http://www.wired.com/wired/archive/4.10/atm.html
[2] http://www.cs.virginia.edu/~mngroup/projects/mpls/documents/thesis/node8.html
[3] http://www.ietf.org/rfc/rfc3439.txt

Connection-oriented communication

Connection-oriented (**CO-mode**[1]) **communication** is a data communication mode in telecommunications whereby the devices at the end points use a protocol to establish an end-to-end logical or physical connection before any data may be sent. In case of digital transmission, in-order delivery of a bit stream or byte stream is provided. Connection-oriented protocol services are often but not always *reliable* network services, that provide acknowledgment after successful delivery, and automatic repeat request functions in case of missing data or detected bit-errors.

Circuit mode communication, for example the public switched telephone network, ISDN, SONET/SDH and optical mesh networks, are examples of connection-oriented communication. Circuit mode communication provides guarantees that data will arrive with constant bandwidth and at constant delay.

Packet mode communication may also be connection-oriented, which is called virtual circuit mode communication. Due to the packet switching, the communication may suffer from variable bit rate and delay, due to varying traffic load and packet queue lengths.

A connection-oriented transport layer protocol, such as TCP, may be based on a connectionless network layer protocol (such as IP), but still achieve in-order delivery of a byte-stream, by means of segment numbering on the sender side and data packet reordering on the receiver side.

In a connection-oriented packet switched data link layer or network layer protocol, all data is sent over the same path during a communication session. The protocol does not have to provide each packet with routing information (complete source and destination address), but only with a channel/data stream number, often denoted virtual circuit identifier (VCI). Routing information may be provided to the network nodes during the connection establishment phase, where the VCI is defined in tables in each node. Thus, the actual packet switching and data transfer can be taken care of by fast hardware, as opposed to slow software based routing.

The alternative to connection-oriented transmission is connectionless communication, also known as datagram communication, in which data is sent in form of packets from one end point to another without prior arrangement or signaling. This mode operates without guarantees of delivery according to a best-effort policy. Each data packet must contain complete address information, since packets are routed individually and independently of each other, possibly transmitted along different network paths.

Connectionless protocols are usually described as stateless because the end points have no protocol-defined way to remember where they are in a "conversation" of message exchanges. Because they can keep track of a conversation, connection-oriented protocols are sometimes described as stateful.

Examples of connection-oriented packet mode communication, i.e. virtual circuit mode communication:

- The Transmission Control Protocol (TCP) is a connection-oriented reliable protocol that is based on a datagram protocol (the IP protocol).
- X.25 is a connection-oriented reliable network protocol.
- Frame relay is a connection-oriented unreliable data link layer protocol.
- GPRS
- Asynchronous Transfer Mode
- Multiprotocol Label Switching

References

[1] Information Processing Systems - Open Systems Interconnection, "Transport Service Definition", International Organization for Standardization, International Standard 8072, June 1986.

Connectionless communication

In packet switching networks, **CL-mode**[1] or **connectionless communication** is a data transmission method in which each data packet carries information in a header record that contains a destination address sufficient to permit the independent delivery of the packet to its destination via the network.

A packet transmitted in a connectionless mode is frequently called a datagram.

In connection-oriented communication the communicating peers must first establish a logical or physical data channel or *connection* in a dialog preceding the exchange of user data.

The connectionless communication mode has the advantage over a connection-oriented mode in that it has low overhead. It also allows for multicast and broadcast operations, which may save even more network resources when the same data needs to be transmitted to several recipients. In contrast, a connection is always unicast (point-to-point).

Unfortunately, in connectionless mode transmission of a packet, the service provider usually cannot guarantee that there will be no loss, error insertion, misdelivery, duplication, or out-of-sequence delivery of the packet. (However, the risk of these hazards may be reduced by providing a reliable transmission service at a higher protocol layer of the OSI Reference Model.)

Another drawback of the connectionless mode is that no optimisations are possible when sending several frames between the same two peers. By establishing a connection at the beginning of such a data exchange the components (routers, bridges) along the network path would be able to pre-compute (and hence cache) routing-related information, avoiding re-computation for every packet. Network components could also reserve capacity for the transfer of the subsequent frames of e.g. a video download.

The distinction between connectionless and connection-oriented transmission may take place at several layers of the OSI Reference Model:

- At the Transport Layer: TCP is a connection-oriented transport protocol. UDP is connectionless.
- At the Network Layer.
- At the Data Link Layer: The IEEE 802.2 protocol at the Logical Link Control sublayer of the data link layer may provide both connectionless and connection-oriented services. In fact, some network protocols (such as SNA's Path Control in its early stages) require a connection-oriented data link layer. Others (like IP) do not. (After the appearance of APPN, SNA could operate on a connectionless data link service as well.)

References

[1] Information Processing Systems - Open Systems Interconnection, "Transport Service Definition - Addendum 1: Connectionless-mode Transmission", International Organization for Standardization, International Standard 8072/AD 1, December 1986.

- *This article incorporates public domain material from the General Services Administration* document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm).

Cyclic redundancy check

A **cyclic redundancy check** (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

CRCs are so called because the *check* (data verification) value is a *redundancy* (it adds no information to the message) and the algorithm is based on *cyclic* codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function. The CRC was invented by W. Wesley Peterson in 1961; the 32-bit polynomial used in the CRC function of Ethernet and many other standards is the work of several researchers and was published in 1975.

Introduction

CRCs are based on the theory of cyclic error-correcting codes. The use of systematic cyclic codes, which encode messages by adding a fixed-length check value, for the purpose of error detection in communication networks was first proposed by W. Wesley Peterson in 1961.[1] Cyclic codes are not only simple to implement but have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages. This is important because burst errors are common transmission errors in many communication channels, including magnetic and optical storage devices. Typically an n-bit CRC applied to a data block of arbitrary length, will detect any single error burst not longer than n bits and will detect a fraction $1-2^{-n}$ of all longer error bursts.

Specification of a CRC code requires definition of a so-called generator polynomial. This polynomial resembles the divisor in a polynomial long division, which takes the message as the dividend and in which the quotient is discarded and the remainder becomes the result, with the important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field. The length of the remainder is always less than the length of the generator polynomial, which therefore determines how long the result can be.

In practice, all commonly used CRCs employ the finite field GF(2). This is the field of two elements, usually called 0 and 1, comfortably matching computer architecture. The rest of this article will discuss only these binary CRCs, but the principles are more general.

The simplest error-detection system, the parity bit, is in fact a trivial 1-bit CRC: it uses the generator polynomial $x+1$.

Application

A CRC-enabled device calculates a short, fixed-length binary sequence, known as the *check value* or improperly *the CRC*, for each block of data to be sent or stored and appends it to the data, forming a *codeword*. When a codeword is received or read, the device either compares its check value with one freshly calculated from the data block, or equivalently, performs a CRC on the whole codeword and compares the resulting check value with an expected *residue* constant. If the check values do not match, then the block contains a *data error* and the device may take corrective action such as rereading or requesting the block be sent again, otherwise the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking).[2]

CRCs and data integrity

CRCs are specifically designed to protect against common types of errors on communication channels, where they can provide quick and reasonable assurance of the integrity of messages delivered. However, they are not suitable for protecting against intentional alteration of data. Firstly, as there is no authentication, an attacker can edit a message and recalculate the CRC without the substitution being detected. This is even the case when the CRC is encrypted, leading to one of the design flaws of the WEP protocol.[3] Secondly, the linear properties of CRC codes even allow an attacker to modify a message in such a way as to leave the check value unchanged,[4] [5] and otherwise permit efficient recalculation of the CRC for compact changes. Nonetheless, it is still often falsely assumed that when a message and its correct check value are received from an open channel then the message cannot have been altered in transit.[6]

Cryptographic hash functions, while still not providing security against intentional alteration when used in this manner, can provide stronger error checking in that they do not rely on specific error pattern assumptions. However, they are much slower than CRCs, and are therefore commonly used to protect off-line data, such as files on servers or databases.

When stored alongside the data, CRCs and cryptographic hash functions by themselves do not protect against *intentional* modification of data. Any application that requires protection against such attacks must use cryptographic authentication mechanisms, such as message authentication codes.

Computation of CRC

To compute an *n*-bit binary CRC, line the bits representing the input in a row, and position the (*n*+1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row.

Start with the message to be encoded:

```
11010011101100
```

This is first padded with zeroes corresponding to the bit length *n* of the CRC. Here is the first calculation for computing a 3-bit CRC:

```
11010011101100 000 <--- input left shifted by 3 bits
1011               <--- divisor (4 bits) = x³+x+1
------------------
01100011101100 000 <--- result
```

If the input bit above the leftmost divisor bit is 0, do nothing. If the input bit above the leftmost divisor bit is 1, the divisor is XORed into the input (in other words, the input bit above each 1-bit in the divisor is toggled). The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the entire calculation:

```
11010011101100 000 <--- input left shifted by 3 bits
1011               <--- divisor
01100011101100 000 <--- result
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000
       1011
```

```
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------
00000000000000 100 <---remainder (3 bits)
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These *n* bits are the remainder of the division step, and will also be the value of the CRC function (unless the chosen CRC specification calls for some postprocessing).

The validity of a received message can easily be verified by performing the above calculation again, this time with the check value added instead of zeroes. The remainder should equal zero if there are no detectable errors.

```
11010011101100 100 <--- input with check value
1011               <--- divisor
01100011101100 100 <--- result
 1011              <--- divisor ...
00111011101100 100



......



00000000001110 100
          1011
00000000000101 100
           101 1
------------------
                 0 <--- remainder
```

Mathematics of CRC

Mathematical analysis of this division-like process reveals how to pick a divisor that guarantees good error-detection properties. In this analysis, the digits of the bit strings are thought of as the coefficients of a polynomial in some variable *x*—coefficients that are elements of the finite field GF(2) instead of more familiar numbers. This binary polynomial is treated as a ring. A ring is, loosely speaking, a set of elements somewhat like numbers, that can be operated on by an operation that somewhat resembles addition and another operation that somewhat resembles multiplication, these operations possessing many of the familiar arithmetic properties of commutativity, associativity, and distributivity. Ring theory is part of abstract algebra.

Designing CRC polynomials

The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error detecting capabilities while minimizing overall collision probabilities.

The most important attribute of the polynomial is its length (largest degree(exponent) +1 of any one term in the polynomial), because of its direct influence on the length of the computed check value.

The most commonly used polynomial lengths are:

- 9 bits (CRC-8)
- 17 bits (CRC-16)
- 33 bits (CRC-32)
- 65 bits (CRC-64)

The design of the CRC polynomial depends on the maximum total length of the block to be protected (data + CRC bits), the desired error protection features, and the type of resources for implementing the CRC as well as the desired performance. A common misconception is that the "best" CRC polynomials are derived from either an irreducible polynomial or an irreducible polynomial times the factor $(1+x)$, which adds to the code the ability to detect all errors affecting an odd number of bits.[7] In reality, all the factors described above should enter in the selection of the polynomial. However, choosing a non-irreducible polynomial can result in missed errors due to the ring having zero divisors.

The advantage of choosing a primitive polynomial as the generator for a CRC code is that the resulting code has maximal total block length; in here if r is the degree of the primitive generator polynomial then the maximal total blocklength is equal to $2^r - 1$, and the associated code is able to detect any single bit or double errors.[8] If instead, we used as generator polynomial $g(x) = p(x)(1+x)$, where $p(x)$ is a primitive polynomial of degree $r-1$, then the maximal total blocklength would be equal to $2^{r-1} - 1$ but the code would be able to detect single, double, and triple errors.

A polynomial $g(x)$ that admits other factorizations may be chosen then so as to balance the maximal total blocklength with a desired error detection power. A powerful class of such polynomials, which subsumes the two examples described above, is that of BCH codes. Regardless of the reducibility properties of a generator polynomial of degree r, assuming that it includes the "+1" term, such error detection code will be able to detect all error patterns that are confined to a window of r contiguous bits. These patterns are called "error bursts".

Specification of CRC

The concept of the CRC as an error-detecting code gets complicated when an implementer or standards committee turns it into a practical system. Here are some of the complications:

- Sometimes an implementation **prefixes a fixed bit pattern** to the bitstream to be checked. This is useful when clocking errors might insert 0-bits in front of a message, an alteration that would otherwise leave the check value unchanged.
- Sometimes an implementation **appends n 0-bits** (n being the size of the CRC) to the bitstream to be checked before the polynomial division occurs. This has the convenience that the remainder of the original bitstream with the check value appended is exactly zero, so the CRC can be checked simply by performing the polynomial division on the received bitstream and comparing the remainder with zero.
- Sometimes an implementation **exclusive-ORs a fixed bit pattern** into the remainder of the polynomial division.
- **Bit order:** Some schemes view the low-order bit of each byte as "first", which then during polynomial division means "leftmost", which is contrary to our customary understanding of "low-order". This convention makes sense when serial-port transmissions are CRC-checked in hardware, because some widespread serial-port transmission conventions transmit bytes least-significant bit first.

- **Byte order**: With multi-byte CRCs, there can be confusion over whether the byte transmitted first (or stored in the lowest-addressed byte of memory) is the least-significant byte or the most-significant byte. For example, some 16-bit CRC schemes swap the bytes of the check value.
- **Omission of the high-order bit** of the divisor polynomial: Since the high-order bit is always 1, and since an *n*-bit CRC must be defined by an (*n*+1)-bit divisor which overflows an *n*-bit register, some writers assume that it is unnecessary to mention the divisor's high-order bit.
- **Omission of the low-order bit** of the divisor polynomial: Since the low-order bit is always 1, authors such as Philip Koopman represent polynomials with their high-order bit intact, but without the low-order bit (the x^0 or 1 term). This convention encodes the polynomial complete with its degree in one integer.

These complications mean that there are three common ways to express a polynomial as an integer: the first two, which are mirror images in binary, are the constants found in code; the third is the number found in Koopman's papers. *In each case, one term is omitted.* So the polynomial $x^4 + x + 1$ may be transcribed as:

- 0x3 = 0011b, representing $x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0$ (MSB-first code)
- 0xC = 1100b, representing $1x^0 + 1x^1 + 0x^2 + 0x^3 + x^4$ (LSB-first code)
- 0x9 = 1001b, representing $1x^4 + 0x^3 + 0x^2 + 1x^1 + x^0$ (Koopman notation)

In the table below they are shown as:

Representations: normal / reversed / reverse of reciprocal
0x3 / 0xC / 0x9

Commonly used and standardized CRCs

Numerous varieties of cyclic redundancy check have been incorporated into technical standards. By no means does one algorithm, or one of each degree, suit every purpose; Koopman and Chakrabarty recommend selecting a polynomial according to the application requirements and the expected distribution of message lengths.[9] The number of distinct CRCs in use have however led to confusion among developers which authors have sought to address.[7] There are three polynomials reported for CRC-12,[9] sixteen conflicting definitions of CRC-16, and six of CRC-32.[10]

The polynomials commonly applied are not the most efficient ones possible. Between 1993 and 2004, Koopman, Castagnoli and others surveyed the space of polynomials up to 16 bits,[9] and of 24 and 32 bits,[11] [12] finding examples that have much better performance (in terms of Hamming distance for a given message size) than the polynomials of earlier protocols, and publishing the best of these with the aim of improving the error detection capacity of future standards.[12] In particular, iSCSI and SCTP have adopted one of the findings of this research, the CRC-32C (Castagnoli) polynomial.

The design of the 32-bit polynomial most commonly used by standards bodies, CRC-32-IEEE, was the result of a joint effort for the Rome Laboratory and the Air Force Electronic Systems Division by Joseph Hammond, James Brown and Shyan-Shiang Liu of the Georgia Institute of Technology and Kenneth Brayer of the MITRE Corporation. The earliest known appearances of the 32-bit polynomial were in their 1975 publications: Technical Report 2956 by Brayer for MITRE, published in January and released for public dissemination through DTIC in August,[13] and Hammond, Brown and Liu's report for the Rome Laboratory, published in May.[14] Both reports contained contributions from the other team. In December 1975, Brayer and Hammond presented their work in a paper at the IEEE National Telecommunications Conference: the IEEE CRC-32 polynomial is the generating polynomial of a Hamming code and was selected for its error detection performance.[15] Even so, the Castagnoli CRC-32C polynomial used in iSCSI or SCTP matches its performance on messages from 58 bits–131 kbits, and outperforms it in several size ranges including the two most common sizes of Internet packet.[12] The ITU-T G.hn standard also uses CRC-32C to detect errors in the payload (although it uses CRC-16-CCITT for PHY headers).

The table below lists only the polynomials of the various algorithms in use. Variations of a particular protocol can impose pre-inversion, post-inversion and reversed bit ordering as described above. For example, the CRC32 used in both Gzip and Bzip2 use the same polynomial, but Bzip2 employs reversed bit ordering, while Gzip does not.

CRCs in proprietary protocols might use a non-trivial initial value and final XOR for obfuscation but this does not add cryptographic strength to the algorithm. An unknown error-detecting code can be characterized as a CRC, and as such fully reverse-engineered, from its output codewords.[16]

Polynomial
ost hardware; also known as *parity bit*)
$+ 1$ (ITU-T G.704 [17], p. 12)
$+ 1$ (Gen 2 RFID[18])
$+ x^2 + 1$ (ITU-T G.704 [17], p. 9)
$+ 1$ (USB token packets)
$+ 1$ (ITU-T G.704 [17], p. 3)
$+ 1$ (telecom systems, ITU-T G.707 [19], ITU-T G.832 [20], MMC, SD)
$+ x + 1$ (ATM HEC), ISDN Header Error Control and Cell Delineation ITU-T I.432.1 (02/99) [21]
$+ x^4 + 1$ (1-Wire bus)
$+ x^6 + x^4 + x^2 + 1$
$+ x^3 + x^2 + 1$
$+ x^4 + x^3 + x + 1$[22]
$^9 + x^5 + x^4 + x + 1$ (ATM; ITU-T I.610 [23])
$^9 + x^8 + x^7 + x^2 + 1$ (FlexRay[24])
$^{11} + x^3 + x^2 + x + 1$ (telecom systems[25] [26])
$^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
$^{15} + x^2 + 1$ (Bisync, Modbus, USB, ANSI X3.28 [27], many others; also known as *CRC-16* and *CRC-16-ANSI*)
$^{12} + x^5 + 1$ (X.25, V.41, HDLC, XMODEM, Bluetooth, SD, many others; known as *CRC-CCITT*)
$^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (SCSI DIF)
$^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$ (DNP, IEC 870, M-Bus)
$^{10} + x^8 + x^7 + x^3 + 1$ (cordless telephones)[29]
ee Fletcher's checksum

$^{2}+x^{20}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^{8}+x^{7}+x^{6}+x^{3}+x+1$	(FlexRay[24])
$^{23}+x^{18}+x^{17}+x^{14}+x^{11}+x^{10}+x^{7}+x^{6}+x^{5}+x^{4}+x^{3}+x+1$	(OpenPGP)
$^{29}+x^{21}+x^{20}+x^{15}+x^{13}+x^{12}+x^{11}+x^{8}+x^{7}+x^{6}+x^{2}+x+1$	(CDMA)
ee Adler-32	
$^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$	(ISO 3309, ANSI X3.66, FIPS PUB 71, FED-STD-1003, ITU-T V.42, Ethernet, SATA, MPEG-2, Gzip, PKZIP, POSIX cksum, PN
$^{3}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1$	(iSCSI & SCTP, G.hn payload, SSE4.2)
$^{0}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+x^{11}+x^{10}+x^{7}+x^{6}+x^{4}+x^{2}+x+1$	
$^{31}+x^{24}+x^{22}+x^{16}+x^{14}+x^{8}+x^{7}+x^{5}+x^{3}+x+1$	(aviation; AIXM[31])
$^{26}+x^{23}+x^{17}+x^{3}+1$	(GSM control channel[32] [33])
$^{4}+x^{3}+x+1$	(HDLC — ISO 3309, Swiss-Prot/TrEMBL; considered weak for hashing[34])
$^{2}+x^{57}+x^{55}+x^{54}+x^{53}+x^{52}+x^{47}+x^{46}+x^{45}+x^{40}+x^{39}+x^{38}+x^{37}+x^{35}+x^{33}+x^{32}+x^{31}+x^{29}+x^{27}+x^{24}+x^{23}+x^{22}+x^{21}+x^{19}+x^{17}+x^{13}+x^{12}+x^{10}+x^{9}+x^{7}+x^{4}+$	ECMA-182[35] p. 51)

References

[1] Peterson, W. W. and Brown, D. T. (January 1961). "Cyclic Codes for Error Detection". *Proceedings of the IRE* **49**: 228. doi:10.1109/JRPROC.1961.287814.

[2] Ritter, Terry (February 1986). "The Great CRC Mystery" (http://www.ciphersbyritter.com/ARTS/CRCMYST.HTM). *Dr. Dobb's Journal* **11** (2): 26–34, 76–83. . Retrieved 21 May 2009.

[3] N. Cam-Winget, Nancy; R. Housley, Russ; D. Wagner, David; J. Walker, Jesse (May 2003). "Security Flaws in 802.11 Data Link Protocols". *Communications of the ACM* **46** (5): 35–39. doi:10.1145/769800.769823.

[4] Stigge, Martin; Plötz, Henryk; Müller, Wolf; Redlich, Jens-Peter (May 2006). *Reversing CRC – Theory and Practice* (http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05_.pdf). Berlin: Humboldt University Berlin. p. 17. . Retrieved 4 February 2011. "The presented methods offer a very easy and efficient way to modify your data so that it will compute to a CRC you want or at least know in advance. This is not a very difficult task, as CRC is not a cryptographical hash algorithm [...] So you should *never* consider the CRC as some kind of message authentication code [...] – it can easily be forged."

[5] Anachriz (30 April 1999). "CRC and how to Reverse it" (http://www.woodmann.com/fravia/crctut1.htm). . Retrieved 21 January 2010. Online essay with example x86 assembly code.

[6] "Eurocontrol – FAQ: Technologies" (http://www.eurocontrol.int/aim/public/faq/chain_faq3.html). European Organisation for the Safety of Air Navigation. . Retrieved 29 April 2009. "A Cyclic Redundancy Check (CRC) is a means by which a data item may be assessed to verify that it has not been changed (either intentionally or unintentionally) since it the CRC value [*sic*] was applied to it."

[7] Williams, Ross N. (24 September 1996). "A Painless Guide to CRC Error Detection Algorithms V3.00" (http://www.repairfaq.org/filipg/LINK/F_crc_v3.html). . Retrieved 5 June 2010.Contains a rigorous explanation of how to generate the CRC table typically found in

implementations.

[8] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 22.4 Cyclic Redundancy and Other Checksums" (http://apps.nrbook.com/empanel/index.html#pg=1168). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8. .

[9] Koopman, Philip; Chakravarty, Tridib (June 2004). "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks" (http://www.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf). *The International Conference on Dependable Systems and Networks*: 145–154. doi:10.1109/DSN.2004.1311885. . Retrieved 14 January 2011.

[10] Cook, Greg (17 October 2011). "Catalogue of parametrised CRC algorithms" (http://regregex.bbcmicro.net/crc-catalogue.htm). . Retrieved 17 October 2011.

[11] Castagnoli, G.; Bräuer, S.; Herrmann, M. (June 1993). "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits". *IEEE Transactions on Communications* **41** (6): 883. doi:10.1109/26.231911. Castagnoli's *et al.* work on algorithmic selection of CRC polynomials.

[12] Koopman, Philip (July 2002). "32-Bit Cyclic Redundancy Codes for Internet Applications" (http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf). *The International Conference on Dependable Systems and Networks*: 459–468. doi:10.1109/DSN.2002.1028931. . Retrieved 14 January 2011. Verification of Castagnoli's results by exhaustive search and some new good polynomials.

[13] Brayer, Kenneth (August 1975). *Evaluation of 32 Degree Polynomials in Error Detection on the SATIN IV Autovon Error Patterns* (http://www.dtic.mil/srch/doc?collection=t3&id=ADA014825). National Technical Information Service. p. 74. . Retrieved 3 February 2011.

[14] Hammond, Joseph L., Jr.; Brown, James E.; Liu, Shyan-Shiang (May 1975). *Development of a Transmission Error Model and an Error Control Model*. National Technical Information Service. p. 74. Bibcode 1975STIN...7615344H. doi:100.2/ADA013939.

[15] Brayer, Kenneth; Hammond, Joseph L., Jr. (December 1975). "Evaluation of error detection polynomial performance on the AUTOVON channel". *Conference Record*. **1**. IEEE National Telecommunications Conference, New Orleans, La. New York: Institute of Electrical and Electronics Engineers. pp. 8–21 to 8–25. Bibcode 1975ntc.....1....8B.

[16] Ewing, Gregory C. (March 2010). "Reverse-Engineering a CRC Algorithm" (http://www.cosc.canterbury.ac.nz/greg.ewing/essays/CRC-Reverse-Engineering.html). Christchurch: University of Canterbury. . Retrieved 26 July 2011.

[17] http://www.itu.int/rec/T-REC-G.704-199810-I/en

[18] *Class-1 Generation-2 UHF RFID Protocol* (http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_1_0_9-standard-20050126.pdf). 1.2.0. EPCglobal. 23 October 2008. p. 35. . Retrieved 21 May 2009.

[19] http://www.itu.int/rec/T-REC-G.707/en

[20] http://www.itu.int/rec/T-REC-G.832/en

[21] http://www.itu.int/rec/T-REC-I.432.1-199902-I/en

[22] Richardson, Andrew (17 March 2005). *WCDMA Handbook*. Cambridge, UK: Cambridge University Press. p. 223. ISBN 0521828155.

[23] http://www.itu.int/rec/T-REC-I.610/en

[24] *FlexRay Protocol Specification*. 2.1 Revision A. Flexray Consortium. 22 December 2005. p. 93.

[25] Perez, A.; Wismer & Becker (1983). "Byte-Wise CRC Calculations". *IEEE Micro* **3** (3): 40–50. doi:10.1109/MM.1983.291120.

[26] Ramabadran, T.V.; Gaitonde, S.S. (1988). "A tutorial on CRC computations". *IEEE Micro* **8** (4): 62–75. doi:10.1109/40.7773.

[27] http://www.incits.org/press/1997/pr97020.htm

[28] Thaler, Pat (28 August 2003). *16-bit CRC polynomial selection* (http://www.t10.org/ftp/t10/document.03/03-290r0.pdf). INCITS T10. . Retrieved 11 August 2009.

[29] *ETSI EN 300 175-3*. V2.2.1. Sophia Antipolis, France: European Telecommunications Standards Institute. November 2008.

[30] Boutell, Thomas; Randers-Pehrson, Glenn; *et al.* (14 July 1998). "PNG (Portable Network Graphics) Specification, Version 1.2" (http://www.libpng.org/pub/png/spec/1.2/PNG-Structure.html). . Retrieved 3 February 2011.

[31] *AIXM Primer* (http://www.eurocontrol.int/aim/gallery/content/public/aicm_aixm_4_5/aixm_primer/AIXM_Primer_4.5.pdf). 4.5. European Organisation for the Safety of Air Navigation. 20 March 2006. . Retrieved 29 April 2009.

[32] Gammel, B.M. (31 October 2005). "Crypto - Codes" (http://users.physik.tu-muenchen.de/gammel/matpack/html/LibDoc/Crypto/MpCRC.html). . Retrieved 10 February 2011.

[33] Geremia, Patrick (April 1999). *Cyclic redundancy check computation: an implementation using the TMS320C54x* (http://focus.ti.com/lit/an/spra530/spra530.pdf). Texas Instruments. p. 5. . Retrieved 10 February 2011.

[34] Jones, David T.. *An Improved 64-bit Cyclic Redundancy Check for Protein Sequences* (http://www.cs.ucl.ac.uk/staff/d.jones/crcnote.pdf). University College London. . Retrieved 15 December 2009.

[35] http://www.ecma-international.org/publications/standards/Ecma-182.htm

External links

- MathPages - Cyclic Redundancy Checks (http://www.mathpages.com/home/kmath458.htm): overview with an explanation of error-detection of different polynomials.
- Clean C++ CRC32 Source Code (http://www.networkdls.com/Software.Asp?Review=22); OS and Library Independent
- The CRC Pitstop (http://www.ross.net/crc/) - home of A Painless Guide to CRC Error Detection Algorithms (http://www.ross.net/crc/crcpaper.html)
- Black, R. (1994-02) Fast CRC32 in Software (http://www.cl.cam.ac.uk/Research/SRG/bluebook/21/crc/crc.html); algorithm 4 is used in Linux and info-zip's zip and unzip.
- Kounavis, M. and Berry, F. (2005). A Systematic Approach to Building High Performance, Software-based, CRC generators (http://www.intel.com/technology/comms/perfnet/download/CRC_generators.pdf), Slicing-by-4 and slicing-by-8 algorithms
- pycrc (http://www.tty1.net/pycrc/) - a free C/C++ source code generator for various CRC algorithms
- CRC32: Generating a checksum for a file (http://www.codeproject.com/KB/recipes/crc32.aspx), C++ implementation by Brian Friesen
- The CRC++ Project (http://www.aweiler.com/crc/) Implementation in C++ which uses template classes to deal with different bit order
- 'CRC-Analysis with Bitfilters'. (http://einstein.informatik.uni-oldenburg.de/papers/CRC-BitfilterEng.pdf)
- Cyclic Redundancy Check (http://www.hackersdelight.org/crc.pdf): theory, practice, hardware, and software with emphasis on CRC-32. A sample chapter from Henry S. Warren, Jr. *Hacker's Delight*.
- CRCs (http://mdfs.net/Info/Comp/Comms/CRCs.htm) Code for CRC-32 and CRC-16 calculation in BASIC, 6502, Z80, PDP-11, 6809, 80x86, ARM and C.
- Reverse-Engineering a CRC Algorithm (http://www.cosc.canterbury.ac.nz/greg.ewing/essays/CRC-Reverse-Engineering.html)

Data link layer

The **data link layer** is **layer 2** of the seven-layer OSI model of computer networking. It corresponds to, or is part of the link layer of the TCP/IP reference model.

The data link layer is the protocol layer that transfers data between adjacent network nodes in a wide area network or between nodes on the same local area network segment.[1] The data link layer provides the functional and procedural means to transfer data between network entities and might provide the means to detect and possibly correct errors that may occur in the physical layer. Examples of data link protocols are Ethernet for local area networks (multi-node), the Point-to-Point Protocol (PPP), HDLC and ADCCP for point-to-point (dual-node) connections.

The data link layer is concerned with local delivery of frames between devices on the same LAN. Data-link frames, as these protocol data units are called, do not cross the boundaries of a local network. Inter-network routing and global addressing are higher layer functions, allowing data-link protocols to focus on local delivery, addressing, and media arbitration. In this way, the data link layer is analogous to a neighborhood traffic cop; it endeavors to arbitrate between parties contending for access to a medium.

When devices attempt to use a medium simultaneously, frame collisions occur. Data-link protocols specify how devices detect and recover from such collisions, and may provide mechanisms to reduce or prevent them.

Delivery of frames by layer-2 devices is effected through the use of unambiguous hardware addresses. A frame's header contains source and destination addresses that indicate which device originated the frame and which device is expected to receive and process it. In contrast to the hierarchical and routable addresses of the network layer, layer-2 addresses are flat, meaning that no part of the address can be used to identify the logical or physical group to which the address belongs.

The data link thus provides data transfer across the physical link. That transfer can be reliable or unreliable; many data-link protocols do not have acknowledgments of successful frame reception and acceptance, and some data-link protocols might not even have any form of checksum to check for transmission errors. In those cases, higher-level protocols must provide flow control, error checking, and acknowledgments and retransmission.

In some networks, such as IEEE 802 local area networks, the data link layer is described in more detail with media access control (MAC) and logical link control (LLC) sublayers; this means that the IEEE 802.2 LLC protocol can be used with all of the IEEE 802 MAC layers, such as Ethernet, token ring, IEEE 802.11, etc., as well as with some non-802 MAC layers such as FDDI. Other data-link-layer protocols, such as HDLC, are specified to include both sublayers, although some other protocols, such as Cisco HDLC, use HDLC's low-level framing as a MAC layer in combination with a different LLC layer. In the ITU-T G.hn standard, which provides a way to create a high-speed (up to 1 Gigabit/s) Local area network using existing home wiring (power lines, phone lines and coaxial cables), the data link layer is divided into three sub-layers (application protocol convergence, logical link control and medium access control).

Within the semantics of the OSI network architecture, the data-link-layer protocols respond to service requests from the network layer and they perform their function by issuing service requests to the physical layer.

Models of communication

Sublayers of the data link layer

The data link layer has two sublayers: *logical link control* (LLC) and *media access control* (MAC).[2]

Logical link control sublayer

The uppermost sublayer, LLC, multiplexes protocols running atop the data link layer, and optionally provides flow control, acknowledgment, and error notification. The LLC provides addressing and control of the data link. It specifies which mechanisms are to be used for addressing stations over the transmission medium and for controlling the data exchanged between the originator and recipient machines.

Media access control sublayer

MAC may refer to the sublayer that determines who is allowed to access the media at any one time (usually CSMA/CD). Other times it refers to a frame structure with MAC addresses inside.

There are generally two forms of media access control: distributed and centralized. Both of these may be compared to communication between people. In a network made up of people speaking, i.e. a conversation, we look for clues from our fellow talkers to see if any of them appear to be about to speak. If two people speak at the same time, they will back off and begin a long and elaborate game of saying "no, you first".

The Media Access Control sublayer also determines where one frame of data ends and the next one starts – frame synchronization. There are four means of frame synchronization: time based, character counting, byte stuffing and bit stuffing.

- The *time based* approach simply puts a specified amount of time between frames. The major drawback of this is that new gaps can be introduced or old gaps can be lost due to external influences.
- *Character counting* simply notes the count of remaining characters in the frame's header. This method, however, is easily disturbed if this field gets faulty in some way, thus making it hard to keep up synchronization.
- *Byte stuffing* precedes the frame with a special byte sequence such as DLE STX and succeeds it with DLE ETX. Appearances of DLE (byte value 0x10) have to be escaped with another DLE. The start and stop marks are detected at the receiver and removed as well as the inserted DLE characters.
- Similarly, *bit stuffing* replaces these start and end marks with flag consisting of a special bit pattern (e.g. a 0, six 1 bits and a 0). Occurrences of this bit pattern in the data to be transmitted is avoided by inserting a bit. To use the example where the flag is 01111110, a 0 is inserted after 5 consecutive 1's in the data stream. The flags and the inserted 0's are removed at the receiving end. This makes for arbitrary long frames and easy synchronization for the recipient. Note that this stuffed bit is added even if the following data bit is 0, which could not be mistaken for a sync sequence, so that the receiver can unambiguously distinguish stuffed bits from normal bits.

List of data-link-layer services

- Encapsulation of network layer data packets into frames
- Frame synchronization
- Logical link control (LLC) sublayer:
  - Error control (automatic repeat request,ARQ), in addition to ARQ provided by some transport-layer protocols, to forward error correction (FEC) techniques provided on the physical layer, and to error-detection and packet canceling provided at all layers, including the network layer. Data-link-layer error control (i.e. retransmission of erroneous packets) is provided in wireless networks and V.42 telephone network modems, but not in LAN protocols such as Ethernet, since bit errors are so uncommon in short wires. In that case, only error detection and canceling of erroneous packets are provided.

  - Flow control, in addition to the one provided on the transport layer. Data-link-layer error control is not used in LAN protocols such as Ethernet, but in modems and wireless networks.
- Media access control (MAC) sublayer:
  - Multiple access protocols for channel-access control, for example CSMA/CD protocols for collision detection and retransmission in Ethernet bus networks and hub networks, or the CSMA/CA protocol for collision avoidance in wireless networks.
  - Physical addressing (MAC addressing)
  - LAN switching (packet switching) including MAC filtering and spanning tree protocol
  - Data packet queueing or scheduling
  - Store-and-forward switching or cut-through switching
  - Quality of Service (QoS) control
  - Virtual LANs (VLAN)

Protocol examples

- Address Resolution Protocol (ARP)
- ARCnet
- ATM
- Cisco Discovery Protocol (CDP)
- Controller Area Network (CAN)
- Econet
- Ethernet
- Ethernet Automatic Protection Switching (EAPS)
- Fiber Distributed Data Interface (FDDI)
- Frame Relay
- High-Level Data Link Control (HDLC)
- IEEE 802.2 (provides LLC functions to IEEE 802 MAC layers)
- IEEE 802.11 wireless LAN
- LattisNet
- Link Access Procedures, D channel (LAPD)
- LocalTalk
- Multiprotocol Label Switching (MPLS)
- Nortel Discovery Protocol (NDP)
- Point-to-Point Protocol (PPP)
- Serial Line Internet Protocol (SLIP) (obsolete)
- Spanning Tree Protocol
- StarLan
- Token ring
- Unidirectional Link Detection (UDLD)
- and most forms of serial communication.

Interfaces

The data link layer is often implemented in software as a "network card driver". The operating system will have a defined software interface between the data link and the network transport stack above. This interface is not a layer itself, but rather a definition for interfacing between layers.

Relation to TCP/IP model

In the frame work of the TCP/IP (Internet Protocol Suite) model, OSI's data link layer, in addition to other components, is contained in TCP/IP's lowest layer, the link layer. The Internet Protocol's link layer only concerns itself with hardware issues to the point of obtaining hardware addresses for locating hosts on a physical network link and transmitting data frames onto the link. Thus, the link layer is broader in scope and encompasses all methods that affect the local link, which is the group of connections that are limited in scope to other nodes on the local access network.

The TCP/IP model is not a top/down comprehensive design reference for networks. It was formulated for the purpose of illustrating the logical groups and scopes of functions needed in the design of the suite of internetworking protocols of TCP/IP, as needed for the operation of the Internet. In general, direct or strict comparisons of the OSI and TCP/IP models should be avoided, because the layering in TCP/IP is not a principal design criterion and in general considered to be "harmful" (RFC 3439). In particular, TCP/IP does not dictate a strict hierarchical sequence of encapsulation requirements, as is attributed to OSI protocols.

References

[1] "What is layer 2, and Why Should You Care?" (http://www.accel-networks.com/blog/2009/09/what-is-layer-2-and-why-should-you-care.html). accel-networks.com. . Retrieved 2009-09-29.

[2] Regis J. Bates and Donald W. Gregory (2007). *Voice & data communications handbook* (http://books.google.com/books?id=eq1kRHdyXSUC&pg=PA45) (5th ed.). McGraw-Hill Professional. p. 45. ISBN 9780072263350. .

External links

- "DataLink layer simulation in C#." (http://www.codeproject.com/KB/IP/DataLink_Simulator.aspx)

Error detection and correction

In information theory and coding theory with applications in computer science and telecommunication, **error detection and correction** or **error control** are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data.

Definitions

The general definitions of the terms are as follows:

- *Error detection* is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.[1]
- *Error correction* is the detection of errors and reconstruction of the original, error-free data.

Introduction

The general idea for achieving error detection and correction is to add some redundancy (i.e., some extra data) to a message, which receivers can use to check consistency of the delivered message, and to recover data determined to be erroneous. Error-detection and correction schemes can be either systematic or non-systematic: In a systematic scheme, the transmitter sends the original data, and attaches a fixed number of *check bits* (or *parity data*), which are derived from the data bits by some deterministic algorithm. If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits; if the values do not match, an error has occurred at some point during the transmission. In a system that uses a non-systematic code, the original message is transformed into an encoded message that has at least as many bits as the original message.

Good error control performance requires the scheme to be selected based on the characteristics of the communication channel. Common channel models include memory-less models where errors occur randomly and with a certain probability, and dynamic models where errors occur primarily in bursts. Consequently, error-detecting and correcting codes can be generally distinguished between *random-error-detecting/correcting* and *burst-error-detecting/correcting*. Some codes can also be suitable for a mixture of random errors and burst errors.

If the channel capacity cannot be determined, or is highly varying, an error-detection scheme may be combined with a system for retransmissions of erroneous data. This is known as automatic repeat request (ARQ), and is most notably used in the Internet. An alternate approach for error control is hybrid automatic repeat request (HARQ), which is a combination of ARQ and error-correction coding.

Implementation

Error correction may generally be realized in two different ways:

- *Automatic repeat request (ARQ)* (sometimes also referred to as *backward error correction*): This is an error control technique whereby an error detection scheme is combined with requests for retransmission of erroneous data. Every block of data received is checked using the error detection code used, and if the check fails, retransmission of the data is requested – this may be done repeatedly, until the data can be verified.
- *Forward error correction (FEC)*: The sender encodes the data using an *error-correcting code (ECC)* prior to transmission. The additional information (redundancy) added by the code is used by the receiver to recover the original data. In general, the reconstructed data is what is deemed the "most likely" original data.

ARQ and FEC may be combined, such that minor errors are corrected without retransmission, and major errors are corrected via a request for retransmission: this is called *hybrid automatic repeat-request (HARQ)*.

Error detection schemes

Error detection is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length *tag* to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided.

There exists a vast variety of different hash function designs. However, some are of particularly widespread use because of either their simplicity or their suitability for detecting certain kinds of errors (e.g., the cyclic redundancy check's performance in detecting burst errors).

Random-error-correcting codes based on minimum distance coding can provide a suitable alternative to hash functions when a strict guarantee on the minimum number of errors to be detected is desired. Repetition codes, described below, are special cases of error-correcting codes: although rather inefficient, they find applications for both error correction and detection due to their simplicity.

Repetition codes

A **repetition code** is a coding scheme that repeats the bits across a channel to achieve error-free communication. Given a stream of data to be transmitted, the data is divided into blocks of bits. Each block is transmitted some predetermined number of times. For example, to send the bit pattern "1011", the four-bit block can be repeated three times, thus producing "1011 1011 1011". However, if this twelve-bit pattern was received as "1010 1011 1011" – where the first block is unlike the other two – it can be determined that an error has occurred.

Repetition codes are very inefficient, and can be susceptible to problems if the error occurs in exactly the same place for each group (e.g., "1010 1010 1010" in the previous example would be detected as correct). The advantage of repetition codes is that they are extremely simple, and are in fact used in some transmissions of numbers stations.

Parity bits

A **parity bit** is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

Extensions and variations on the parity bit mechanism are horizontal redundancy checks, vertical redundancy checks, and "double," "dual," or "diagonal" parity (used in RAID-DP).

Checksums

A **checksum** of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a ones'-complement operation prior to transmission to detect errors resulting in all-zero messages.

Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers.

Cyclic redundancy checks (CRCs)

A **cyclic redundancy check (CRC)** is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a so-called *generator polynomial*, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result.

Cyclic codes have favorable properties in that they are well suited for detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives.

Even parity is a special case of a cyclic redundancy check, where the single-bit CRC is generated by the divisor $x + 1$.

Cryptographic hash functions

A **cryptographic hash function** can provide strong assurances about data integrity, provided that changes of the data are only accidental (i.e., due to transmission errors). Any modification to the data will likely be detected through a mismatching hash value. Furthermore, given some hash value, it is infeasible to find some input data (other than the one given) that will yield the same hash value. Message authentication codes, also called *keyed* cryptographic hash functions, provide additional protection against intentional modification by an attacker.

Error-correcting codes

Any error-correcting code can be used for error detection. A code with *minimum Hamming distance*, d, can detect up to $d - 1$ errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired.

Codes with minimum Hamming distance $d = 2$ are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code.

The Berger code is an early example of a unidirectional error(-correcting) code that can detect any number of errors on an asymmetric channel, provided that only transitions of cleared bits to set bits *or* set bits to cleared bits can occur.

Error correction

Automatic repeat request

Automatic Repeat reQuest (ARQ) is an error control method for data transmission that makes use of error-detection codes, acknowledgment and/or negative acknowledgment messages, and timeouts to achieve reliable data transmission. An *acknowledgment* is a message sent by the receiver to indicate that it has correctly received a data frame.

Usually, when the transmitter does not receive the acknowledgment before the timeout occurs (i.e., within a reasonable amount of time after sending the data frame), it retransmits the frame until it is either correctly received or the error persists beyond a predetermined number of retransmissions.

Three types of ARQ protocols are Stop-and-wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ.

ARQ is appropriate if the communication channel has varying or unknown capacity, such as is the case on the Internet. However, ARQ requires the availability of a back channel, results in possibly increased latency due to retransmissions, and requires the maintenance of buffers and timers for retransmissions, which in the case of network congestion can put a strain on the server and overall network capacity.[2]

Error-correcting code

An error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or *parity data*, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

Error-correcting codes are usually distinguished between convolutional codes and block codes:

- *Convolutional codes* are processed on a bit-by-bit basis. They are particularly suitable for implementation in hardware, and the Viterbi decoder allows optimal decoding.
- *Block codes* are processed on a block-by-block basis. Early examples of block codes are repetition codes, Hamming codes and multidimensional parity-check codes. They were followed by a number of efficient codes, Reed-Solomon codes being the most notable due to their current widespread use. Turbo codes and low-density parity-check codes (LDPC) are relatively new constructions that can provide almost optimal efficiency.

Shannon's theorem is an important theorem in forward error correction, and describes the maximum information rate at which reliable communication is possible over a channel that has a certain error probability or signal-to-noise ratio (SNR). This strict upper limit is expressed in terms of the channel capacity. More specifically, the theorem says that there exist codes such that with increasing encoding length the probability of error on a discrete memoryless channel can be made arbitrarily small, provided that the code rate is smaller than the channel capacity. The code rate is defined as the fraction k/n of k source symbols and n encoded symbols.

The actual maximum code rate allowed depends on the error-correcting code used, and may be lower. This is because Shannon's proof was only of existential nature, and did not show how to construct codes which are both optimal and have efficient encoding and decoding algorithms.

Hybrid schemes

Hybrid ARQ is a combination of ARQ and forward error correction. There are two basic approaches:[2]

- Messages are always transmitted with FEC parity data (and error-detection redundancy). A receiver decodes a message using the parity information, and requests retransmission using ARQ only if the parity data was not sufficient for successful decoding (identified through a failed integrity check).
- Messages are transmitted without parity data (only with error-detection information). If a receiver detects an error, it requests FEC information from the transmitter using ARQ, and uses it to reconstruct the original message.

The latter approach is particularly attractive on an erasure channel when using a rateless erasure code.

Applications

Applications that require low latency (such as telephone conversations) cannot use Automatic Repeat reQuest (ARQ); they must use Forward Error Correction (FEC). By the time an ARQ system discovers an error and re-transmits it, the re-sent data will arrive too late to be any good.

Applications where the transmitter immediately forgets the information as soon as it is sent (such as most television cameras) cannot use ARQ; they must use FEC because when an error occurs, the original data is no longer available. (This is also why FEC is used in data storage systems such as RAID and distributed data store).

Applications that use ARQ must have a return channel. Applications that have no return channel cannot use ARQ.

Applications that require extremely low error rates (such as digital money transfers) must use ARQ.

The Internet

In a typical TCP/IP stack, error control is performed at multiple levels:

- Each Ethernet frame carries a CRC-32 checksum. Frames received with incorrect checksums are discarded by the receiver hardware.
- The IPv4 header contains a checksum protecting the contents of the header. Packets with mismatching checksums are dropped within the network or at the receiver.
- The checksum was omitted from the IPv6 header in order to minimize processing costs in network routing and because current link layer technology is assumed to provide sufficient error detection (see also RFC 3819).
- UDP has an optional checksum covering the payload and addressing information from the UDP and IP headers. Packets with incorrect checksums are discarded by the operating system network stack. The checksum is optional under IPv4, only, because the Data-Link layer checksum may already provide the desired level of error protection.
- TCP provides a checksum for protecting the payload and addressing information from the TCP and IP headers. Packets with incorrect checksums are discarded within the network stack, and eventually get retransmitted using ARQ, either explicitly (such as through triple-ack) or implicitly due to a timeout.

Deep-space telecommunications

Development of error-correction codes was tightly coupled with the history of deep-space missions due to the extreme dilution of signal power over interplanetary distances, and the limited power availability aboard space probes. Whereas early missions sent their data uncoded, starting from 1968 digital error correction was implemented in the form of (sub-optimally decoded) convolutional codes and Reed-Muller codes.[3] The Reed-Muller code was well suited to the noise the spacecraft was subject to (approximately matching a bell curve), and was implemented at the Mariner spacecraft for missions between 1969 and 1977.

The Voyager 1 and Voyager 2 missions, which started in 1977, were designed to deliver color imaging amongst scientific information of Jupiter and Saturn.[4] This resulted in increased coding requirements, and thus the spacecraft were supported by (optimally Viterbi-decoded) convolutional codes that could be concatenated with an outer Golay (24,12,8) code. The Voyager 2 probe additionally supported an implementation of a Reed-Solomon code: the concatenated Reed-Solomon-Viterbi (RSV) code allowed for very powerful error correction, and enabled the spacecraft's extended journey to Uranus and Neptune.

The CCSDS currently recommends usage of error correction codes with performance similar to the Voyager 2 RSV code as a minimum. Concatenated codes are increasingly falling out of favor with space missions, and are replaced by more powerful codes such as Turbo codes or LDPC codes.

The different kinds of deep space and orbital missions that are conducted suggest that trying to find a "one size fits all" error correction system will be an ongoing problem for some time to come. For missions close to earth the nature of the channel noise is different from that of a spacecraft on an interplanetary mission experiences. Additionally, as a spacecraft increases its distance from earth, the problem of correcting for noise gets larger.

Satellite broadcasting (DVB)

The demand for satellite transponder bandwidth continues to grow, fueled by the desire to deliver television (including new channels and High Definition TV) and IP data. Transponder availability and bandwidth constraints have limited this growth, because transponder capacity is determined by the selected modulation scheme and Forward error correction (FEC) rate.

Overview

- QPSK coupled with traditional Reed Solomon and Viterbi codes have been used for nearly 20 years for the delivery of digital satellite TV.
- Higher order modulation schemes such as 8PSK, 16QAM and 32QAM have enabled the satellite industry to increase transponder efficiency by several orders of magnitude.
- This increase in the information rate in a transponder comes at the expense of an increase in the carrier power to meet the threshold requirement for existing antennas.
- Tests conducted using the latest chipsets demonstrate that the performance achieved by using Turbo Codes may be even lower than the 0.8 dB figure assumed in early designs.

Data storage

Error detection and correction codes are often used to improve the reliability of data storage media.

A "parity track" was present on the first magnetic tape data storage in 1951. The "Optimal Rectangular Code" used in group code recording tapes not only detects but also corrects single-bit errors.

Some file formats, particularly archive formats, include a checksum (most often CRC32) to detect corruption and truncation and can employ redundancy and/or parity files to recover portions of corrupted data.

Reed Solomon codes are used in compact discs to correct errors caused by scratches.

Modern hard drives use CRC codes to detect and Reed-Solomon codes to correct minor errors in sector reads, and to recover data from sectors that have "gone bad" and store that data in the spare sectors.[5]

RAID systems use a variety of error correction techniques, to correct errors when a hard drive completely fails.

Error-correcting memory

DRAM memory may provide increased protection against soft errors by relying on error correcting codes. Such error-correcting memory, known as *ECC* or *EDAC-protected* memory, is particularly desirable for high fault-tolerant applications, such as servers, as well as deep-space applications due to increased radiation.

Error-correcting memory controllers traditionally use Hamming codes, although some use triple modular redundancy.

Interleaving allows distributing the effect of a single cosmic ray potentially upsetting multiple physically neighboring bits across multiple words by associating neighboring bits to different words. As long as a single event upset (SEU) does not exceed the error threshold (e.g., a single error) in any particular word between accesses, it can be corrected (e.g., by a single-bit error correcting code), and the illusion of an error-free memory system may be maintained.[6]

Notes

[1] See also Encyclopedia of Microcomputers: Volume 6 (http://books.google.com.br/books?id=ajpoqYqFrkQC&lpg=PA343&dq=Error detection and correction is&as_brr=3&client=firefox-a&pg=PA343#v=snippet&q="error detection is"&f=false), p. 343.

[2] A. J. McAuley, *Reliable Broadband Communication Using a Burst Erasure Correcting Code*, ACM SIGCOMM, 1990.

[3] K. Andrews et al., *The Development of Turbo and LDPC Codes for Deep-Space Applications*, Proceedings of the IEEE, Vol. 95, No. 11, Nov. 2007.

[4] W. Huffman, V. Pless, *Fundamentals of error-correcting codes*, Cambridge University Press, ISBN 9780521782807, 2003.

[5] My Hard Drive Died | Scott A. Moulton (http://www.myharddrivedied.com/presentations_whitepaper.html)

[6] "Using StrongArm SA-1110 in the On-Board Computer of Nanosatellite" (http://www.apmcsta.org/File/doc/Conferences/6th meeting/Chen Zhenyu.doc). Tsinghua Space Center, Tsinghua University, Beijing. . Retrieved 2009-02-16.

References

- Shu Lin, Daniel J. Costello, Jr. (1983). *Error Control Coding: Fundamentals and Applications*. Prentice Hall. ISBN 0-13-283796-X.

External links

- The on-line textbook: Information Theory, Inference, and Learning Algorithms (http://www.inference.phy.cam.ac.uk/mackay/itila/), by David MacKay, contains chapters on elementary error-correcting codes; on the theoretical limits of error-correction; and on the latest state-of-the-art error-correcting codes, including low-density parity-check codes, turbo codes, and fountain codes.
- Compute parameters of linear codes (http://ipsit.bu.edu/comp.html) - an on-line interface for generating and computing parameters (e.g. minimum distance, covering radius) of linear error-correcting codes.
- ECC Page (http://www.eccpage.com/)

Ethernet

Ethernet /ˈiːθərnɛt/ is a family of computer networking technologies for local area networks (LANs) commercially introduced in 1980. Standardized in IEEE 802.3, Ethernet has largely replaced competing wired LAN technologies.

An 8P8C modular connector (often called RJ45) commonly used on cat 5 cables in Ethernet networks

Systems communicating over Ethernet divide a stream of data into individual packets called frames. Each frame contains source and destination addresses and error-checking data so that damaged data can be detected and re-transmitted.

The standards define several wiring and signaling variants. The original 10BASE5 Ethernet used coaxial cable as a shared medium. Later the coaxial cables were replaced by twisted pair and fiber optic links in conjunction with hubs or switches. Data rates were periodically increased from the original 10 megabits per second, to 100 gigabits per second.

Since its commercial release, Ethernet has retained a good degree of compatibility. Features such as the 48-bit MAC address and Ethernet frame format have influenced other networking protocols.

History

Ethernet was developed at Xerox PARC between 1973 and 1974.[1] [2] It was inspired by ALOHAnet, which Robert Metcalfe had studied as part of his PhD dissertation.[3] The idea was first documented in a memo that Metcalfe wrote on May 22, 1972.[1] [4] In 1975, Xerox filed a patent application listing Metcalfe, David Boggs, Chuck Thacker and Butler Lampson as inventors.[5] In 1976, after the system was deployed at PARC, Metcalfe and Boggs published a seminal paper.[6] [7]

Metcalfe left Xerox in June 1979 to form 3Com.[1] He convinced Digital Equipment Corporation (DEC), Intel, and Xerox to work together to promote Ethernet as a standard. The so-called "DIX" standard, for "Digital/Intel/Xerox" specified 10 Mbit/s Ethernet, with 48-bit destination and source addresses and a global 16-bit Ethertype-type field. It was published on September 30, 1980 as "The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications".[8] Version 2 was published in November, 1982[9] and defines what has become known as Ethernet II. Formal standardization efforts proceeded at the same time.

Ethernet initially competed with two largely proprietary systems, Token Ring and Token Bus. Because Ethernet was able to adapt to market realities and shift to inexpensive and ubiquitous twisted pair wiring, these proprietary protocols soon found themselves competing in a market inundated by Ethernet products and by the end of the 1980s, Ethernet was clearly the dominant network technology.[1] In the process, 3Com became a major company. 3Com shipped its first 10 Mbit/s Ethernet 3C100 transceiver in March 1981, and that year started selling adapters for PDP-11s and VAXes, as well as Multibus-based Intel and Sun Microsystems computers.[10] :9 This was followed quickly by DEC's Unibus to Ethernet adapter, which DEC sold and used internally to build its own corporate network, which reached over 10,000 nodes by 1986, making it one of the largest computer networks in the world at that time.[11]

Since then Ethernet technology has evolved to meet new bandwidth and market requirements.[12] In addition to computers, Ethernet is now used to interconnect appliances and other personal devices.[1] It is used in industrial applications and is quickly replacing legacy data transmission systems in the world's telecommunications networks.[13] By 2010, the market for Ethernet equipment amounted to over $16 billion per year.[14]

Standardization

Notwithstanding its technical merits, timely standardization was instrumental to the success of Ethernet. It required well-coordinated and partly competitive activities in several standardization bodies such as the IEEE, ECMA, IEC, and finally ISO.

In February 1980, the Institute of Electrical and Electronics Engineers (IEEE) started project 802 to standardize local area networks (LAN).[15]

The "DIX-group" with Gary Robinson (DEC), Phil Arst (Intel), and Bob Printis (Xerox) submitted the so-called "Blue Book" CSMA/CD specification as a candidate for the LAN specification.[8] Since IEEE membership is open to all professionals, including students, the group received countless comments on this technology.

In addition to CSMA/CD, Token Ring (supported by IBM) and Token Bus (selected and henceforward supported by General Motors) were also considered as candidates for a LAN standard. Due to the goal of IEEE 802 to forward only one standard and due to the strong company support for all three designs, the necessary agreement on a LAN standard was significantly delayed.

In the Ethernet camp, it put at risk the market introduction of the Xerox Star workstation and 3Com's Ethernet LAN products. With such business implications in mind, David Liddle (General Manager, Xerox Office Systems) and Metcalfe (3Com) strongly supported a proposal of Fritz Röscheisen (Siemens Private Networks) for an alliance in the emerging office communication market, including Siemens' support for the international standardization of Ethernet (April 10, 1981). Ingrid Fromm, Siemens' representative to IEEE 802, quickly achieved broader support for Ethernet beyond IEEE by the establishment of a competing Task Group "Local Networks" within the European

standards body ECMA TC24. As early as March 1982 ECMA TC24 with its corporate members reached agreement on a standard for CSMA/CD based on the IEEE 802 draft. The speedy action taken by ECMA decisively contributed to the conciliation of opinions within IEEE and approval of IEEE 802.3 CSMA/CD by the end of 1982. IEEE published the 802.3 standard as a draft in 1983[10] :8 and as a standard in 1985.

Approval of Ethernet on the international level was achieved by a similar, cross-partisan action with Fromm as liaison officer working to integrate International Electrotechnical Commission, TC83 and International Organization for Standardization (ISO) TC97SC6, and the ISO/IEEE 802/3 standard was approved in 1984.

Evolution

Ethernet evolved to include higher bandwidth, improved media access control methods, and different physical media. The coaxial cable was replaced with point-to-point links connected by Ethernet repeaters or switches to reduce installation costs, increase reliability, and improve management and troubleshooting. Many variants of Ethernet remain in common use.

Ethernet stations communicate by sending each other data packets: blocks of data individually sent and delivered. As with other IEEE 802 LANs, each Ethernet station is given a 48-bit MAC address. The MAC addresses are used to specify both the destination and the source of each data packet. Ethernet establishes link level connections, which can be defined using both the destination and sources addresses. On reception of a transmission, the receiver uses the destination address to determine whether the transmission is relevant to the station or should be ignored. Network interfaces normally do not accept packets addressed to other Ethernet stations. Adapters come programmed with a globally unique address.[16] An Ethertype field in each frame is used by the operating system on the receiving station to select the appropriate protocol module (i.e. the Internet protocol module). Ethernet frames are said to be *self-identifying*, because of the frame type. Self-identifying frames make it possible to intermix multiple protocols on the same physical network and allow a single computer to use multiple protocols together.[17] Despite the significant changes in Ethernet, all generations of Ethernet (excluding early experimental versions) use the same frame formats[18] (and hence the same interface for higher layers), and can be readily interconnected through bridging.

Due to the ubiquity of Ethernet, the ever-decreasing cost of the hardware needed to support it, and the reduced panel space needed by twisted pair Ethernet, most manufacturers now build Ethernet interfaces directly into PC motherboards, eliminating the need for installation of a separate network card.[19]

Shared media

Ethernet was originally based on the idea of computers communicating over a shared coaxial cable acting as a broadcast transmission medium. The methods used were similar to those used in radio systems,[20] with the common cable providing the communication channel likened to the *Luminiferous aether* in 19th century physics, and it was from this reference that the name "Ethernet" was derived.[21]

10BASE5 Ethernet equipment

Original Ethernet's shared coaxial cable (the shared medium) traversed a building or campus to every attached machine. A scheme known as carrier sense multiple access with collision detection (CSMA/CD) governed the way the computers shared the channel. This scheme was simpler than the competing token ring or token bus technologies.[22] Computers were connected to an Attachment Unit Interface (AUI) transceiver, which was in turn connected to the cable (later with thin

Ethernet the transceiver was integrated into the network adapter). While a simple passive wire was highly reliable for small networks, it was not reliable for large extended networks, where damage to the wire in a single place, or a single bad connector, could make the whole Ethernet segment unusable.[23]

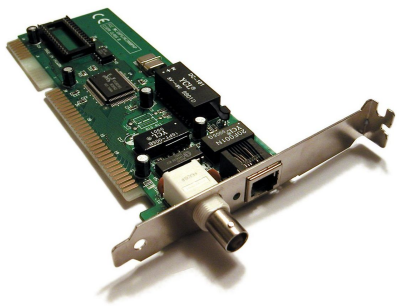
Through the first half of the 1980s, Ethernet's 10BASE5 implementation used a coaxial cable 0.375 inches (9.5 mm) in diameter, later called "thick Ethernet" or "thicknet". Its successor, 10BASE2, called "thin Ethernet" or "thinnet", used a cable similar to cable television cable of the era. The emphasis was on making installation of the cable easier and less costly.

Since all communications happen on the same wire, any information sent by one computer is received by all, even if that information is intended for just one destination.[24] The network interface card interrupts the CPU only when applicable packets are received: The card ignores information not addressed to it.[25] Use of a single cable also means that the bandwidth is shared, so that network traffic can be very slow when many stations are simultaneously active.

Collisions corrupt transmitted data and require stations to retransmit. The lost data and retransmissions reduce throughput. In the worst case, when there are lots of hosts with long cables that attempt to transmit many short frames, excessive collisions can reduce throughput dramatically. However, a Xerox report in 1980 summarized the results of having 20 fast nodes attempting to transmit packets of various sizes as quickly as possible on the same Ethernet segment.[26] The results showed that, even for the smallest Ethernet frames (64 bytes), 90% throughput on the LAN was the norm. This is in comparison with token passing LANs (token ring, token bus), all of which suffer throughput degradation as each new node comes into the LAN, due to token waits. This report was controversial, as modeling showed that collision-based networks became unstable under loads as low as 40% of nominal capacity. Many early researchers failed to understand the subtleties of the CSMA/CD protocol and how important it was to get the details right, and were really modeling somewhat different networks (usually not as good as real Ethernet).[27]

Repeaters and hubs

For signal degradation and timing reasons, coaxial Ethernet segments had a restricted size. Somewhat larger networks could be built by using an Ethernet repeater. Early repeaters had only two ports, allowing, at most, a doubling of network size. Once repeaters with 4, 6, 8, and more ports became available, it was possible to wire the network in a star topology. Early experiments with star topologies (called "Fibernet") using optical fiber were published by 1978.[28]



A 1990s network interface card supporting both coaxial cable-based 10BASE2 (BNC connector, left) and twisted pair-based 10BASE-T (8P8C connector, right)

Shared cable Ethernet was always hard to install in offices because its bus topology was in conflict with the star topology cable plans designed into buildings for telephony. Modifying Ethernet to conform to twisted pair telephone wiring already installed in commercial buildings provided another opportunity to lower costs, expand the installed base, and leverage building design, and, thus, twisted-pair Ethernet was the next logical development in the mid-1980s.

Ethernet on unshielded twisted-pair cables (UTP) began with StarLAN at 1 Mbit/s in the mid-1980s. SynOptics introduced the first twisted-pair Ethernet at 10 Mbit/s in a star-wired cabling topology with a central hub, later called LattisNet.[29] [30] These evolved into 10BASE-T, which was designed for point-to-point links only, and all termination was built into the device. This changed repeaters from a specialist device used at the center of large networks to a device that every twisted pair-based network with more than two machines had to use. The tree structure that resulted from this made Ethernet networks easier to maintain by preventing most faults with one peer or its associated cable from affecting other devices on the network.

Despite the physical star topology and the presence of separate transmit and receive channels in the twisted pair and fiber media, repeater based Ethernet networks still use half-duplex and CSMA/CD, with only minimal activity by the repeater, primarily the Collision Enforcement signal, in dealing with packet collisions. Every packet is sent to every port on the repeater, so bandwidth and security problems are not addressed. The total throughput of the repeater is limited to that of a single link, and all links must operate at the same speed.

Bridging and switching

Ethernet switches

While repeaters could isolate some aspects of Ethernet segments, such as cable breakages, they still forwarded all traffic to all Ethernet devices. This created practical limits on how many machines could communicate on an Ethernet network. The entire network was one collision domain, and all hosts had to be able to detect collisions anywhere on the network. This limited the number of repeaters between the farthest nodes. Segments joined by repeaters had to all operate at the same speed, making phased-in upgrades impossible.

To alleviate these problems, bridging was created to communicate at the data link layer while isolating the physical layer. With bridging, only well-formed Ethernet packets are forwarded from one Ethernet segment to another; collisions and packet errors are isolated. Prior to discovery of network devices on the different segments, Ethernet bridges (and switches) work somewhat like Ethernet repeaters, passing all traffic between segments. However, as the bridge discovers the addresses associated with each port, it forwards network traffic only to the necessary segments, improving overall performance. Broadcast traffic is still forwarded to all network segments. Bridges also overcame the limits on total segments between two hosts and allowed the mixing of speeds, both of which became very important with the introduction of Fast Ethernet.

In 1989, the networking company Kalpana introduced their EtherSwitch, the first Ethernet switch.[31] This worked somewhat differently from an Ethernet bridge, in that only the header of the incoming packet would be examined before it was either dropped or forwarded to another segment. This greatly reduced the forwarding latency and the processing load on the network device. One drawback of this cut-through switching method was that packets that had been corrupted would still be propagated through the network, so a jabbering station could continue to disrupt the entire network. The eventual remedy for this was a return to the original store and forward approach of bridging, where the packet would be read into a buffer on the switch in its entirety, verified against its checksum and then forwarded, but using more powerful application-specific integrated circuits. Hence, the bridging is then done in hardware, allowing packets to be forwarded at full wire speed.

When a twisted pair or fiber link segment is used and neither end is connected to a repeater, full-duplex Ethernet becomes possible over that segment. In full-duplex mode, both devices can transmit and receive to and from each other at the same time, and there is no collision domain. This doubles the aggregate bandwidth of the link and is sometimes advertised as double the link speed (e.g., 200 Mbit/s).[32] The elimination of the collision domain for these connections also means that all the link's bandwidth can be used by the two devices on that segment and that segment length is not limited by the need for correct collision detection.

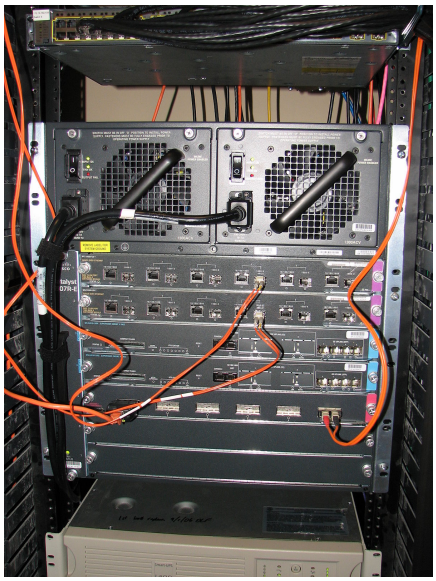
Since packets are typically delivered only to the port they are intended for, traffic on a switched Ethernet is less public than on shared-medium Ethernet. Despite this, switched Ethernet should still be regarded as an insecure network technology, because it is easy to subvert switched Ethernet systems by means such as ARP spoofing and MAC flooding.

The bandwidth advantages, the slightly better isolation of devices from each other, the ability to easily mix different speeds of devices and the elimination of the chaining limits inherent in non-switched Ethernet have made switched

Ethernet the dominant network technology.[33]

Advanced networking

Simple switched Ethernet networks, while a great improvement over repeater-based Ethernet, suffer from single points of failure, attacks that trick switches or hosts into sending data to a machine even if it is not intended for it, scalability and security issues with regard to broadcast radiation and multicast traffic, and bandwidth choke points where a lot of traffic is forced down a single link.



A core Ethernet switch

Advanced networking features in switches and routers combat these issues through means including spanning-tree protocol to maintain the active links of the network as a tree while allowing physical loops for redundancy, port security and protection features such as MAC lock down and broadcast radiation filtering, virtual LANs to keep different classes of users separate while using the same physical infrastructure, multilayer switching to route between different classes and link aggregation to add bandwidth to overloaded links and to provide some measure of redundancy.

Networking advances IEEE 802.1aq (SPB) include the use of the link-state routing protocol IS-IS to allow larger networks with shortest path routes between devices.

Varieties of Ethernet

The Ethernet physical layer evolved over a considerable time span and encompasses quite a few physical media interfaces and several magnitudes of speed. The most common forms used are 10BASE-T, 100BASE-TX, and 1000BASE-T. All three utilize twisted pair cables and 8P8C modular connectors. They run at 10 Mbit/s, 100 Mbit/s, and 1 Gbit/s, respectively. Fiber optic variants of Ethernet offer high performance, electrical isolation and distance (tens of kilometers with some versions). In general, network protocol stack software will work similarly on all varieties.

Ethernet frames

A data packet on the wire is called a frame. A frame begins with preamble and start frame delimiter, followed by an Ethernet header featuring source and destination MAC addresses. The middle section of the frame consists of payload data including any headers for other protocols (e.g., Internet Protocol) carried in the frame. The frame ends with a 32-bit cyclic redundancy check, which is used to detect corruption of data in transit.

Autonegotiation

Autonegotiation is the procedure by which two connected devices choose common transmission parameters, such as speed and duplex mode. Autonegotiation was first introduced as an optional feature for 100BASE-TX, but it is also backward compatible with 10BASE-T. Autonegotiation is mandatory for 1000BASE-T.

Notes

[1] *The History of Ethernet* (http://www.youtube.com/watch?v=g5MezxMcRmk). NetEvents.tv. 2006. . Retrieved September 10, 2011.

[2] "Ethernet Prototype Circuit Board" (http://americanhistory.si.edu/collections/object.cfm?key=35&objkey=96). Smithsonian National Museum of American History. 1973. . Retrieved September 2, 2007.

[3] Gerald W. Brock (September 25, 2003). *The Second Information Revolution*. Harvard University Press. p. 151. ISBN 0674011783.

[4] Mary Bellis. "Inventors of the Modern Computer" (http://inventors.about.com/library/weekly/aa111598.htm). About.com. . Retrieved September 10, 2011.

[5] U.S. Patent 4063220 (http://www.google.com/patents?vid=4063220) "Multipoint data communication system (with collision detection)"

[6] Robert Metcalfe; David Boggs (July 1976). "Ethernet: Distributed Packet Switching for Local Computer Networks" (http://www.acm.org/classics/apr96/). *Communications of the ACM* **19** (7): 395–405. doi:10.1145/360248.360253. .

[7] The experimental Ethernet described in the 1976 paper ran at 2.94 Mbit/s and had eight-bit destination and source address fields, so the original Ethernet addresses were not the MAC addresses they are today.John F. Shoch; Yogen K. Dalal; David D. Redell; Ronald C. Crane (August 1982). "Evolution of the Ethernet Local Computer Network" (http://ethernethistory.typepad.com/papers/EthernetEvolution.pdf). *IEEE Computer* **15** (8): 14–26. doi:10.1109/MC.1982.1654107. . By software convention, the 16 bits after the destination and source address fields specified a "packet type", but, as the paper says, "different protocols use disjoint sets of packet types". Thus the original packet types could vary within each different protocol, rather than the packet type in the current Ethernet standard, which specifies the protocol being used.

[8] Digital Equipment Corporation, Intel Corporation and Xerox Corporation (30 September 1980), *The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications, Version 1.0* (http://ethernethistory.typepad.com/papers/EthernetSpec.pdf), Xerox Corporation, , retrieved 2011-12-10

[9] Digital Equipment Corporation, Intel Corporation and Xerox Corporation (November 1982), *The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications, Version 2.0* (http://decnet.ipv7.net/docs/dundas/aa-k759b-tk.pdf), Xerox Corporation, , retrieved 2011-12-10

[10] Robert Breyer & Sean Riley (1999). *Switched, Fast, and Gigabit Ethernet*. Macmillan. ISBN 1-57870-073-6.

[11] Jamie Parker Pearson (1992). *Digital at Work*. Digital Press. p. 163. ISBN 1555580920.

[12] Rick Merritt (December 20, 2010). *Shifts, growth ahead for 10G Ethernet* (http://www.eetimes.com/electronics-news/4211609/Shifts-growth-ahead-for-10G-Ethernet). E Times. . Retrieved September 10, 2011.

[13] "My oh My – Ethernet Growth Continues to Soar; Surpasses Legacy" (http://www.jaymiescotto.com/jsablog/2011/07/29/my-oh-my-ethernet-growth-continues-to-soar-surpasses-legacy/). Telecom News Now. July 29, 2011. . Retrieved September 10, 2011.

[14] Jim Duffy (February 22, 2010). *Cisco, Juniper, HP drive Ethernet switch market in Q4* (http://www.networkworld.com/news/2010/022210-ethernet-switch-market.html). Network World. . Retrieved September 10, 2011.

[15] Vic Hayes (August 27, 2001). "Letter to FCC" (http://www.ieeeusa.org/policy/policy/2001/01aug27IEEE802.pdf). . Retrieved October 22, 2010. "IEEE 802 has the basic charter to develop and maintain networking standards... IEEE 802 was formed in February 1980..."

[16] In some cases, the factory-assigned address can be overridden, either to avoid an address change when an adapter is replaced or to use locally administered addresses.

[17] Douglas E. Comer (2000). *Internetworking with TCP/IP – Principles, Protocols and Architecture* (4th ed.). Prentice Hall. ISBN 0-13-018380-6. 2.4.9 – Ethernet Hardware Addresses, p. 29, explains the filtering.

[18] Iljitsch van Beijnum. "Speed matters: how Ethernet went from 3Mbps to 100Gbps... and beyond" (http://arstechnica.com/gadgets/news/2011/07/ethernet-how-does-it-work.ars). Ars Technica. . Retrieved July 15, 2011. "All aspects of Ethernet were changed: its MAC procedure, the bit encoding, the wiring... only the packet format has remained the same."

[19] Geetaj Channana (November 1, 2004). "Motherboard Chipsets Roundup" (http://pcquest.ciol.com/content/search/showarticle.asp?artid=63428). PCQuest. . Retrieved October 22, 2010. "While comparing motherboards in the last issue we found that all motherboards support Ethernet connection on board."

[20] There are fundamental differences between wireless and wired shared-medium communications, such as the fact that it is much easier to detect collisions in a wired system than a wireless system.

[21] Charles E. Spurgeon (2000). *Ethernet: The Definitive Guide*. O'Reilly. ISBN 978-1565926608.

[22] In a CSMA/CD system packets must be large enough to guarantee that the leading edge of the propagating wave of the message got to all parts of the medium before the transmitter could stop transmitting, thus guaranteeing that collisions (two or more packets initiated within a window of time that forced them to overlap) would be discovered. Minimum packet size and the physical medium's total length were, thus, closely linked.

[23] Multipoint systems are also prone to strange failure modes when an electrical discontinuity reflects the signal in such a manner that some nodes would work properly, while others work slowly because of excessive retries or not at all. See standing wave for an explanation. These could be much more difficult to diagnose than a complete failure of the segment.

[24] This "one speaks, all listen" property is a security weakness of shared-medium Ethernet, since a node on an Ethernet network can eavesdrop on all traffic on the wire if it so chooses.

[25] Unless it is put into "promiscuous mode"

[26] Shoch, John F. and Hupp, Jon A. (December 1980). "Measured performance of an Ethernet local network" (http://portal.acm.org/citation.cfm?doid=359038.359044#abstract). *Communications of the ACM* (ACM Press) **23** (12): 711–721. doi:10.1145/359038.359044. ISSN 0001-0782. .

[27] Boggs, D.R., Mogul, J.C., and Kent, C.A. (August 1988). "Measured capacity of an Ethernet: myths and reality" (http://portal.acm.org/citation.cfm?doid=52325.52347#abstract). *ACM SIGCOMM Computer Communication Review* (ACM Press) **18** (4): 222–234. doi:10.1145/52325.52347. ISBN 0-89791-279-9. .

[28] Eric G. Rawson; Robert M. Metcalfe (July 1978). "Fibemet: Multimode Optical Fibers for Local Computer Networks" (http://ethernethistory.typepad.com/papers/Fibernet.pdf). *IEEE transactions on communications* **26** (7): 983–990. doi:10.1109/TCOM.1978.1094189. . Retrieved June 11, 2011.

[29] Spurgeon, Charles E. (2000). *Ethernet; The Definitive Guide* (http://books.google.com/books?id=MRChaUQr0Q0C&pg=PA20&lpg=PA20&dq=synoptics+unshielded+twisted+pair&source=bl&ots=oF5HLbKhsN&sig=aw-dUL9TPoDSaZT0I5ztZvchmjE&hl=en&ei=jGSGTfDQNIqDgAex8InECA&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCQQ6AEwADg8#v=onepage&q=synoptics&f=false). Nutshell Handbook. O'Reilly. p. 29. ISBN 1-56592-660-9. .

[30] Urs von Burg (2001). *The Triumph of Ethernet: technological communities and the battle for the LAN standard* (http://books.google.com/books?id=ooBqdIXIqbwC&pg=PA175). Stanford University Press. p. 175. ISBN 0-8047-4094-1. .

[31] The term *switch* was invented by device manufacturers and does not appear in the 802.3 standard.

[32] This is misleading, as performance will double only if traffic patterns are symmetrical.

[33] "Token Ring-to-Ethernet Migration" (http://www.cisco.com/en/US/solutions/collateral/ns340/ns394/ns74/ns149/net_business_benefit09186a00800c92b9_ps6600_Products_White_Paper.html). Cisco. . Retrieved October 22, 2010. "Respondents were first asked about their current and planned desktop LAN attachment standards. The results were clear—switched Fast Ethernet is the dominant choice for desktop connectivity to the network"

References

Further reading

- Digital Equipment Corporation, Intel Corporation, Xerox Corporation (September, 1980). *The Ethernet: A Local Area Network* (http://portal.acm.org/citation.cfm?id=1015591.1015594). — Version 1.0 of the DIX specification.
- "Internetworking Technology Handbook" (http://docwiki.cisco.com/wiki/Ethernet_Technologies). Cisco Systems. Retrieved April 11, 2011.

External links

- IEEE 802.3 Ethernet working group (http://www.ieee802.org/3/)
- IEEE 802.3-2008 standard (http://standards.ieee.org/getieee802/802.3.html)

Ethernet frame

A data packet on an Ethernet link is called an **Ethernet frame**. A frame begins with Preamble and Start Frame Delimiter. Following which, each Ethernet frame continues with an Ethernet header featuring destination and source MAC addresses. The middle section of the frame is payload data including any headers for other protocols (e.g. Internet Protocol) carried in the frame. The frame ends with a 32-bit cyclic redundancy check which is used to detect any corruption of data in transit.

Structure

A data packet on the wire is called a frame and consists of binary data. A frame viewed on the physical wire would show Preamble and Start Frame Delimiter, in addition to the other data. These are required by all physical hardware.[1]

The table below shows the complete Ethernet frame, as transmitted, for the MTU of 1500 octets (some implementations of gigabit Ethernet and higher speeds support larger jumbo frames).[2] One octet is eight bits of data (i.e., a byte on most modern computers).

802.3 Ethernet frame structure

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype or length	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets of 10101010	1 octet of 10101011	6 octets	6 octets	(4 octets)	2 octets	46–1500 octets	4 octets	12 octets
		64–1522 octets						
72–1530 octets								
84–1542 octets								

Note: 2nd row of above table is misleading. It should be refactored. It suggests than *if* optional 802.1Q tag is used then min valid frame length is 68 octets. Its actually still 64 octets. This is because the additional 4 octets used due to the 802.1Q tag is considered to be part of the payload (or in the parlance of the IEEE 802.3-2008 standard, part of MAC CLIENT DATA).

Preamble and Start Frame Delimiter

The Start Frame Delimiter (SFD) is the 8-bit (1-byte) value marking the end of the preamble of an Ethernet frame. The SFD is immediately followed by the destination MAC address. It has the value 10101011. The preamble of an Ethernet frame consists of a 56-bit (7-byte) pattern of alternating 1 and 0 bits, which allows devices on the network to easily detect a new incoming frame. The SFD is designed to break this pattern, and signal the start of the actual frame.

10/100M transceiver chips (MII PHY) work with 4-bits (one nibble) at a time. Therefore the preamble will consist of 7 instances of 0101 + 0101, and the Start Frame Delimiter 0101 + 1101. 8-bit values are sent low 4-bit first and then high 4-bit. 1000M transceiver chips (GMII) work with 8-bits at a time, and 10 Gbit/s (XGMII) PHY works with 32-bits at a time. Note that when using octets, first 7 octets of 01010101 are sent, and then one octet of 11010101. But because the low 4-bit nibble 0101 is sent first, and later the high 4-bit nibble 1101, the Start-of-frame sequence 1101 will be sent after the preamble not before the last 4-bits of the preamble as one might otherwise be led to believe. Don't forget that there are differences between DIX V2.0 (Ethernet 2) & IEEE 802.3 In the Frame format.

Header

The header features source and destination MAC addresses which have 48-bits (6 bytes) each, the Ethertype protocol identifier field and optional IEEE 802.1Q VLAN tag indicating VLAN membership and traffic priority.

EtherType or Length

This is a two-octet (2 bytes) field in an Ethernet frame. It is used to indicate which protocol is encapsulated in the Payload of an Ethernet Frame.

Payload or data of an Ethernet frame

The minimum frame payload is 368 bit(46 Bytes) (dictated by the slot time of the Ethernet LAN architecture). The maximum frame payload is 12000 (1500 Bytes), which is achieved by a single transmitting node which does not therefore suffer

Frame check sequence

The frame check sequence (CRC) is a 32-bit(4 bytes) cyclic redundancy check which enables detection of corrupted data within the entire frame.

Interframe gap

After a frame has been sent, transmitters are required to transmit a minimum of 96 bits (12 octets) of idle line state before transmitting the next frame.

Ethernet frame types

There are several types of Ethernet frames. The different frame types have different formats and MTU values, but can coexist on the same physical medium.

- The Ethernet Version 2[3] or Ethernet II frame or DIX frame is the most common type in use today, as it is often used directly by the Internet Protocol.
- Novell's non-standard variation of raw IEEE 802.3 frame
- IEEE 802.2 Logical Link Control (LLC) frame
- Subnetwork Access Protocol (SNAP) frame

In addition, all four Ethernet frames types may optionally contain a IEEE 802.1Q tag to identify what VLAN it belongs to and its IEEE 802.1p priority (quality of service). This encapsulation is defined in the IEEE 802.3ac specification and increases the maximum frame by 4 bytes.

The IEEE 802.1Q tag, if present, is placed between the Source Address and the EtherType or Length fields. The first two bytes of the tag are the Tag Protocol Identifier (TPID) value of 0x8100. This is located in the same place as the EtherType/Length field in untagged frames, so an EtherType value of 0x8100 means the frame is tagged, and the true EtherType/Length is located after the Q-tag. The TPID is followed by two bytes containing the Tag Control Information (TCI) (the IEEE 802.1p priority (quality of service) and VLAN id). The Q-tag is followed by the rest of the frame, using one of the types described above.

Ethernet II

Ethernet II framing (also known as **DIX Ethernet**, named after DEC, Intel and Xerox, the major participants in its design[4]), defines the two-octet EtherType field in an Ethernet frame, preceded by destination and source MAC addresses, that identifies an upper layer protocol encapsulating the frame data. For example, an EtherType value of 0x0800 signals that the frame contains an IPv4 datagram. Likewise, an EtherType of 0x0806 indicates an ARP frame, 0x8100 indicates an IEEE 802.1Q frame and 0x86DD indicates an IPv6 frame.

As this industry-developed standard went through a formal IEEE standardization process, the EtherType field was changed to a (data) length field in the new 802.3 standard.[5] Since the packet recipient still needs to know how to interpret the packet, the standard required an IEEE 802.2 header to follow the length and specify the packet type. Many years later, the 802.3x-1997 standard, and later versions of the 802.3 standard, formally approved of both types of framing. In practice, both formats are in wide use, with original Ethernet framing the most common in Ethernet local area networks, due to its simplicity and lower overhead.

In order to allow some packets using Ethernet v2 framing and some packets using the original version of 802.3 framing to be used on the same Ethernet segment, EtherType values must be greater than or equal to 1536 (0x0600). That value was chosen because the maximum length of the data field of an Ethernet 802.3 frame is 1500 bytes (0x05DC). Thus if the field's value is greater than or equal to 1536, the frame must be an Ethernet v2 frame, with that field being a type field.[6] If it's less than or equal to 1500, it must be an IEEE 802.3 frame, with that field being a length field. Values between 1500 and 1536, exclusive, are undefined.[7] This convention allows software to determine whether a frame is an Ethernet II frame or an IEEE 802.3 frame, allowing the coexistence of both standards on the same physical medium.

802.2 LLC

Some protocols, particularly those designed for the OSI stack, operate directly on top of IEEE 802.2 LLC encapsulation, which provides both connection-oriented and connectionless network services.

IEEE 802.2 LLC encapsulation is not in widespread use on common networks currently, with the exception of large corporate NetWare installations that have not yet migrated to NetWare over IP. In the past, many corporate networks used IEEE 802.2 to support transparent translating bridges between Ethernet and Token Ring or FDDI networks.

There exists an Internet standard for encapsulating IPv4 traffic in IEEE 802.2 LLC SAP/SNAP frames.[8] It is almost never implemented on Ethernet, although it is used on FDDI, Token Ring, IEEE 802.11, and other IEEE 802 LANs. IP traffic cannot be encapsulated in IEEE 802.2 LLC frames without SNAP because, although there is a LLC SAP protocol type for IP, there is no such type for ARP, which is required for operation of any medium to large network. IPv6 can also be transmitted over Ethernet using IEEE 802.2 LLC SAP/SNAP, but, again, that's almost never used.

SNAP

By examining the 802.2 LLC header, it is possible to determine whether it is followed by a SNAP header. The LLC header includes two additional eight-bit address fields, called *service access points* (SAPs) in OSI terminology; when both source and destination SAP are set to the value 0xAA, the SNAP service is requested. The SNAP header allows EtherType values to be used with all IEEE 802 protocols, as well as supporting private protocol ID spaces. In IEEE 802.3x-1997, the IEEE Ethernet standard was changed to explicitly allow the use of the 16-bit field after the MAC addresses to be used as a length field or a type field.

Mac OS uses IEEE 802.2 LLC SAP/SNAP encapsulation for the AppleTalk v2 protocol suite on Ethernet (“EtherTalk”.)

Novell raw 802.3

Novell's "raw" 802.3 frame format was based on early IEEE 802.3 work. Novell used this as a starting point to create the first implementation of its own IPX Network Protocol over Ethernet. They did not use any LLC header but started the IPX packet directly after the length field. This does not conform to the IEEE 802.3 standard, but since IPX has always FF at the first two bytes (while in IEEE 802.2 LLC that pattern is theoretically possible but extremely unlikely), in practice this mostly coexists on the wire with other Ethernet implementations, with the notable exception of some early forms of DECnet which got confused by this.

Novell NetWare used this frame type by default until the mid nineties, and since Netware was very widespread back then, while IP was not, at some point in time most of the world's Ethernet traffic ran over "raw" 802.3 carrying IPX. Since Netware 4.10, Netware now defaults to IEEE 802.2 with LLC (Netware Frame Type Ethernet_802.2) when using IPX.[9]

Maximum throughput

We may calculate the **protocol efficiency** for Ethernet:

$$\text{Protocol efficiency} = \frac{\text{Payload size}}{\text{Frame size}}$$

Maximum efficiency is achieved with largest allowed payload size and is

$$\frac{1500}{1538} = 97.53\%$$

for untagged Ethernet packets, since the frame size is maximum 1500 byte payload + 8 byte preamble + 14 byte header + 4 Byte trailer + minimum interframe gap corresponding to 12 bytes = 1538 bytes. The maximum efficiency is

$$\frac{1500}{1542} = 97.28\%$$

when 802.1Q VLAN tagging is used.

The throughput may be calculated from the efficiency:

$$\text{Throughput} = \text{Efficiency} \times \text{Net bit rate},$$

where the physical layer net bit rate (the wire bit rate) depends on the Ethernet physical layer standard, and may be 10 Mbit/s, 100 Mbit/s, 1 Gbit/s or 10 Gbit/s. Maximum throughput for 100BASE-TX Ethernet is consequently 97.53 Mbit/s without 802.1Q, and 97.28 Mbit/s with 802.1Q.

Channel utilization is concept often confused with protocol efficiency. It considers only the use of the channel disregarding the nature of the data transmitted – either payload or overhead. At the physical layer, the link channel and equipment do not know the difference between data and control frames. We may calculate the link utilization or channel utilization:

$$\text{Channel utilization} = \frac{\text{Time spent transmitting data}}{\text{Total time}}$$

The total time considers the round trip time along the channel, the processing time in the hosts and the time transmitting data and acknowledge packets. The time spent transmitting data includes data and acknowledge packets.

Runt frames

A runt frame is an Ethernet frame that is less than the IEEE 802.3 minimum length of 64 bytes. Possible causes are collision, underruns, bad network card or software.[10]

Notes

[1] Preamble and Start Frame Delimiter are not displayed by packet sniffing software because these bits are stripped away at OSI Layer 1 by the Ethernet adapter before being passed on to the OSI Layer 2 which is where packet sniffers collect their data from. There are OSI Physical Layer sniffers which can capture and display the Preamble and Start Frame but they are expensive and mainly used to detect physical related problems.

[2] The bit patterns in the preamble and start of frame delimiter are written as bit strings, with the first bit transmitted on the left (*not* as byte values, which in Ethernet are transmitted least significant bit(s) first). This notation matches the one used in the IEEE 802.3 standard.

[3] A version 1 Ethernet frame was used for early Ethernet prototypes and featured 8-bit MAC addresses and was never commercially deployed.

[4] Drew Heywood; Zubair Ahmad (2001). *Drew Heywood's Windows 2000 Network Services*. Sams. p. 53. ISBN 0672317419.

[5] Original Ethernet packets define their length with the framing that surrounds it, rather than with an explicit length count.

[6] LAN MAN Standards Committee of the IEEE Computer Society (20 March 1997). *IEEE Std 802.3x-1997 and IEEE Std 802.3y-1997*. The Institute of Electrical and Electronics Engineers, Inc.. pp. 28–31.

[7] IEEE Std 802.3-2005, 3.2.6

[8] "RFC1042: A Standard for the Transmission of IP Datagrams over IEEE 802 Networks" (http://tools.ietf.org/html/rfc1042). Network Working Group of the IETF. February 1988. .

[9] Don Provan (1993-09-17). "ep17.190654.13335@novell.com Ethernet Framing (news:1993S)". [news:comp.sys.novell comp.sys.novell]. (Web link) (http://groups.google.com/group/bit.listserv.novell/browse_thread/thread/d00a24530625714c). — a classic series of Usenet postings by Novell's Don Provan that have found their way into numerous FAQs and are widely considered the definitive answer to the Novell Frame Type usage.

[10] "Troubleshooting Ethernet" (http://www.cisco.com/en/US/docs/internetworking/troubleshooting/guide/tr1904.html). Cisco Systems. .

References

Ethernet hub

An **Ethernet hub**, **active hub**, **network hub**, **repeater hub** or **hub** is a device for connecting multiple Ethernet devices together and making them act as a single network segment. A hub works at the physical layer (layer 1) of the OSI model.[1] The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

4-port Ethernet hub

Some hubs may also come with a BNC and/or Attachment Unit Interface (AUI) connector to allow connection to legacy 10BASE2 or 10BASE5 network segments. The availability of low-priced network switches has largely rendered hubs obsolete but they are still seen in older installations and more specialized applications.

Technical information

A network hub is an unsophisticated device in comparison with, for example, a switch. A hub does not examine or manage any of the traffic that comes through it: any packet entering any port is rebroadcast on all other ports.[2] Effectively, it is barely aware of frames or packets and mostly operates on raw bits. Consequently, packet collisions are more frequent in networks connected using hubs than in networks connected using more sophisticated devices.[1]

100 Mbit/s hubs and repeaters come in two different speed grades: Class I delay the signal for a maximum of 140 bit times (enabling translation between 100Base-TX, 100Base-FX and 100Base-T4) and Class II hubs delay the signal for a maximum of 92 bit times (enabling installation of two hubs in a single collision domain).[3]

The need for hosts to be able to detect collisions limits the number of hubs and the total size of a network built using hubs (a network built using switches does not have these limitations). For 10 Mbit/s networks built using repeater hubs, the 5-4-3 rule must be followed: up to 5 segments (4 hubs) are allowed between any two end stations.[2] For 10BASE-T networks, up to six segments and five repeaters are allowed between any two hosts. For 100 Mbit/s networks, the limit is reduced to 3 segments (2 hubs) between any two end stations, and even that is only allowed if the hubs are of Class II. Some hubs have manufacturer specific stack ports allowing them to be combined in a way that allows more hubs than simple chaining through Ethernet cables, but even so, a large fast Ethernet network is likely to require switches to avoid the chaining limits of hubs.[1]

Most hubs detect typical problems, such as excessive collisions and jabbering on individual ports, and *partition* the port, disconnecting it from the shared medium. Thus, hub-based twisted-pair Ethernet is generally more robust than coaxial cable-based Ethernet (e.g. 10BASE2), where a misbehaving device can adversely affect the entire collision domain.[2] Even if not partitioned automatically, a hub simplifies troubleshooting because hubs remove the need to troubleshoot faults on a long cable with multiple taps; status lights on the hub can indicate the possible problem source or, as a last resort, devices can be disconnected from a hub one at a time much more easily than from a coaxial cable.

Hubs are classified as physical layer devices in the OSI model. At the physical layer, hubs support little in the way of sophisticated networking. Hubs do not read any of the data passing through them and are not aware of their source or destination addressing. A hub simply receives incoming Ethernet frames, regenerates the electrical signal, and

broadcasts these packets out to all other devices on the network.[1]

To pass data through the repeater in a usable fashion from one segment to the next, the framing and data rate must be the same on each segment. This means that a repeater cannot connect an 802.3 segment (Ethernet) and an 802.5 segment (Token Ring) or a 10 MBit/s segment to 100 MBit/s Ethernet.

Dual-speed hub

In the early days of fast Ethernet, Ethernet switches were relatively expensive devices. Hubs suffered from the problem that if there were any 10BASE-T devices connected then the whole network needed to run at 10 Mbit/s. Therefore a compromise between a hub and a switch was developed, known as a **dual-speed hub**. These devices consisted of an internal two-port switch, dividing the 10 Mbit/s and 100 Mbit/s segments. The device would typically consist of more than two physical ports. When a network device becomes active on any of the physical ports, the device attaches it to either the 10 Mbit/s segment or the 100 Mbit/s segment, as appropriate. This prevented the need for an all-or-nothing migration fast Ethernet networks. These devices are considered hubs because the traffic between devices connected at the same speed is not switched.

Uses

Historically, the main reason for purchasing hubs rather than switches was their price. This motivator has largely been eliminated by reductions in the price of switches, but hubs can still be useful in special circumstances:

- For inserting a protocol analyzer into a network connection, a hub is an alternative to a network tap or port mirroring.[4]
- When a switch is accessible for end users to make connections, for example, in a conference room, an inexperienced or careless user (or saboteur) can bring down the network by connecting two ports together, causing a loop. This can be prevented by using a hub, where a loop will break other users on the hub, but not the rest of the network. This hazard can also be avoided by using switches that can detect and deal with loops, for example by implementing the spanning tree protocol.
- A hub with a 10BASE2 port can be used to connect devices that only support 10BASE2 to a modern network. The same goes for linking in an old 10BASE5 network segment using an AUI port on a hub. Individual devices that were intended for thicknet can also be linked to modern Ethernet by using an AUI-10BASE-T transceiver.

References

[1] Dean, Tamara (2010). *Network+ Guide to Networks*. Delmar. pp. 256–257.

[2] Hallberg, Bruce (2010). *Networking: A Beginner's Guide, Fifth Edition*. McGraw Hill. pp. 68–69.

[3] "What is the difference between Class I and Class II hubs?" (http://www.intel.com/support/express/hubs/sb/CS-020686.htm) Intel. Retrieved 2011-03-16.

[4] "Sniffing Tutorial part 1 - Intercepting Network Traffic" (http://www.netresec.com/?page=Blog&month=2011-03&post=Sniffing-Tutorial-part-1---Intercepting-Network-Traffic). NETRESEC Network Security Blog. 2011-03-11. . Retrieved 2011-03-13.

External links

- Hub Reference (http://wiki.wireshark.org/HubReference)

Frame (networking)

In computer networking and telecommunication, a **frame** is a digital data transmission unit or data packet that includes frame synchronization, i.e. a sequence of bits or symbols making it possible for the receiver to detect the beginning and end of the packet in the stream of symbols or bits. If a receiver is connected to the system in the middle of a frame transmission, it ignores the data until it detects a new frame synchronization sequence.

In computer networking, a frame is a data packet on the Layer 2 of the OSI model.[1] A frame is "the unit of transmission in a link layer protocol, and consists of a link-layer header followed by a packet."[2] Examples are Ethernet frames (maximum 1500 byte plus overhead), PPP frames and V.42 modem frames.

In telecommunications, specifically time-division multiplex (TDM) and time-division multiple access (TDMA), a frame is a cyclically repeated data block that consists of a fixed number of time slots, one for each logical TDM channel or TDMA transmitter. In this consist, a frame is typically an entity at the physical layer. TDM application examples are SONET/SDH and the ISDN circuit switched B-channel. TDMA examples are the 2G and 3G circuit switched cellular voice services. The frame is also an entity for time-division duplex, where the mobile terminal may transmit during some timeslots and receive during others.

References

[1] "Data Link Layer (Layer 2)" (http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm). The TCP/IP Guide. 2005-09-20. . Retrieved 2010-01-31.

[2] *"RFC 1122: Requirements for Internet Hosts — Communication Layers"* (http://tools.ietf.org/html/rfc1122#page-18). IETF. October 1989. p. 18. RFC 1122. . Retrieved 2010-06-07.

Frame Relay

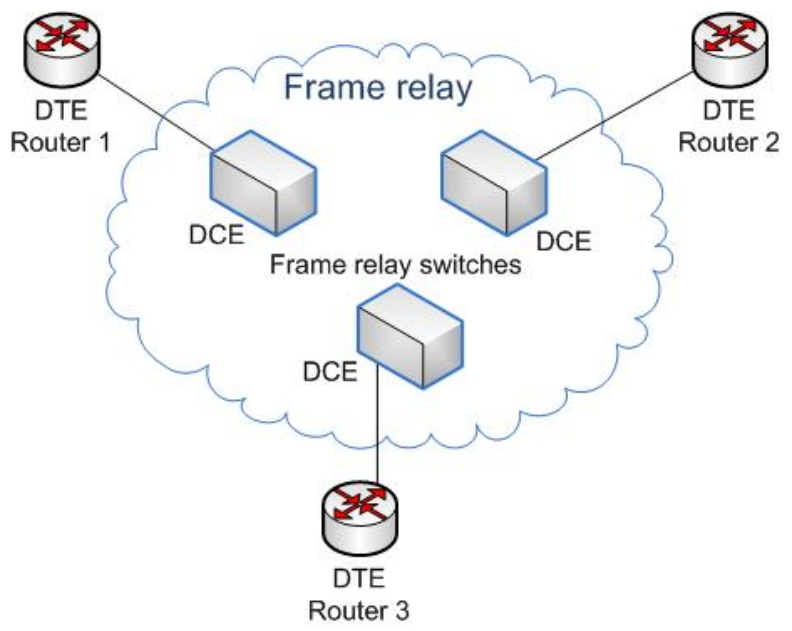
Frame Relay is a standardized wide area network technology that specifies the physical and logical link layers of digital telecommunications channels using a packet switching methodology. Originally designed for transport across Integrated Services Digital Network (ISDN) infrastructure, it may be used today in the context of many other network interfaces.

Network providers commonly implement Frame Relay for voice (VoFR) and data as an encapsulation technique, used between local area networks (LANs) over a wide area network (WAN). Each end-user gets a private line (or leased line) to a Frame Relay node. The Frame Relay network handles the transmission over a frequently-changing path transparent to all end-users.

Frame Relay has become one of the most extensively-used WAN protocols. Its cheapness (compared to leased lines) provided one reason for its popularity. The extreme simplicity of configuring user equipment in a Frame Relay network offers another reason for Frame Relay's popularity.

With the advent of Ethernet over fiber optics, MPLS, VPN and dedicated broadband services such as cable modem and DSL, the end may loom for the Frame Relay protocol and encapsulation. However many rural areas remain lacking DSL and cable modem services. In such cases the least expensive type of non-dial-up connection remains a 64-kbit/s frame-relay line. Thus a retail chain, for instance, may use Frame Relay for connecting rural stores into their corporate WAN.

Technical description



A basic Frame Relay network

The designers of Frame Relay aimed to a telecommunication service for cost-efficient data transmission for intermittent traffic between local area networks (LANs) and between end-points in a wide area network (WAN). Frame Relay puts data in variable-size units called "frames" and leaves any necessary error-correction (such as re-transmission of data) up to the end-points. This speeds up overall data transmission. For most services, the network provides a permanent virtual circuit (PVC), which means that the customer sees a continuous, dedicated connection without having to pay for a full-time leased line, while the service-provider figures out the route each frame travels to its destination and can charge based on usage.

An enterprise can select a level of service quality - prioritizing some frames and making others less important. Frame Relay can run on fractional T-1 or full T-carrier system carriers. Frame Relay complements and provides a mid-range service between basic rate ISDN, which offers bandwidth at 128 kbit/s, and Asynchronous Transfer Mode (ATM), which operates in somewhat similar fashion to frame Relay but at speeds from 155.520 Mbit/s to 622.080 Mbit/s.

Frame Relay has its technical base in the older X.25 packet-switching technology, designed for transmitting data on analog voice lines. Unlike X.25, whose designers expected analog signals, Frame Relay offers a fast packet technology, which means that the protocol does not attempt to correct errors. When a Frame Relay network detects an error in a frame, it simply drops that frame. The end points have the responsibility for detecting and retransmitting dropped frames. (However, digital networks offer an incidence of error extraordinarily small relative to that of analog networks.)

Frame Relay often serves to connect local area networks (LANs) with major backbones as well as on public wide-area networks (WANs) and also in private network environments with leased lines over T-1 lines. It requires a dedicated connection during the transmission period. Frame Relay does not provide an ideal path for voice or video transmission, both of which require a steady flow of transmissions. However, under certain circumstances, voice and video transmission do use Frame Relay.

Frame Relay originated as an extension of Integrated Services Digital Network (ISDN). Its designers aimed to enable a packet-switched network to transport the circuit-switched technology. The technology has become a stand-alone and cost-effective means of creating a WAN.

Frame Relay switches create virtual circuits to connect remote LANs to a WAN. The Frame Relay network exists between a LAN border device, usually a router, and the carrier switch. The technology used by the carrier to transport data between the switches is variable and may differ among carriers (i.e. to function, a practical Frame Relay implementation need not rely solely on its own transportation mechanism).

The sophistication of the technology requires a thorough understanding of the terms used to describe how Frame Relay works. Without a firm understanding of Frame Relay, it is difficult to troubleshoot its performance.

Frame-relay frame structure essentially mirrors almost exactly that defined for LAP-D. Traffic analysis can distinguish Frame Relay format from LAP-D by its lack of a control field.

Protocol data unit

Each Frame Relay Protocol data unit (PDU) consists of the following fields:

1. **Flag Field**. The flag is used to perform high-level data link synchronization which indicates the beginning and end of the frame with the unique pattern 01111110. To ensure that the 01111110 pattern does not appear somewhere inside the frame, bit stuffing and destuffing procedures are used.
2. **Address Field**. Each address field may occupy either octet 2 to 3, octet 2 to 4, or octet 2 to 5, depending on the range of the address in use. A two-octet address field comprises the EA=ADDRESS FIELD EXTENSION BITS and the C/R=COMMAND/RESPONSE BIT.
   1. **DLCI**-Data Link Connection Identifier Bits. The DLCI serves to identify the virtual connection so that the receiving end knows which information connection a frame belongs to. Note that this DLCI has only local significance. A single physical channel can multiplex several different virtual connections.
   2. **FECN, BECN, DE** bits. These bits report congestion:
      - **FECN**=Forward Explicit Congestion Notification bit
      - **BECN**=Backward Explicit Congestion Notification bit
      - **DE**=Discard Eligibility bit
3. **Information Field**. A system parameter defines the maximum number of data bytes that a host can pack into a frame. Hosts may negotiate the actual maximum frame length at call set-up time. The standard specifies the maximum information field size (supportable by any network) as at least 262 octets. Since end-to-end protocols typically operate on the basis of larger information units, Frame Relay recommends that the network support the maximum value of at least 1600 octets in order to avoid the need for segmentation and reassembling by end-users.
4. **Frame Check Sequence (FCS) Field**. Since one cannot completely ignore the bit error-rate of the medium, each switching node needs to implement error detection to avoid wasting bandwidth due to the transmission of *err*ed frames. The error detection mechanism used in Frame Relay uses the cyclic redundancy check (CRC) as its basis.

Congestion control

The Frame Relay network uses a simplified protocol at each switching node. It achieves simplicity by omitting link-by-link flow-control. As a result, the offered load has largely determined the performance of Frame Relay networks. When offered load is high, due to the bursts in some services, temporary overload at some Frame Relay nodes causes a collapse in network throughput. Therefore, frame-relay networks require some effective mechanisms to control the congestion.

Congestion control in frame-relay networks includes the following elements:

1. Admission Control. This provides the principal mechanism used in Frame Relay to ensure the guarantee of resource requirement once accepted. It also serves generally to achieve high network performance. The network decides whether to accept a new connection request, based on the relation of the requested traffic descriptor and the network's residual capacity. The traffic descriptor consists of a set of parameters communicated to the switching nodes at call set-up time or at service-subscription time, and which characterizes the connection's statistical properties. The traffic descriptor consists of three elements:
2. Committed Information Rate (CIR). The average rate (in bit/s) at which the network guarantees to transfer information units over a measurement interval T. This T interval is defined as: T = Bc/CIR.
3. Committed Burst Size (BC). The maximum number of information units transmittable during the interval T.
4. Excess Burst Size (BE). The maximum number of uncommitted information units (in bits) that the network will attempt to carry during the interval.

Once the network has established a connection, the edge node of the Frame Relay network must monitor the connection's traffic flow to ensure that the actual usage of network resources does not exceed this specification. Frame Relay defines some restrictions on the user's information rate. It allows the network to enforce the end user's information rate and discard information when the subscribed access rate is exceeded.

Explicit congestion notification is proposed as the congestion avoidance policy. It tries to keep the network operating at its desired equilibrium point so that a certain Quality of Service (QoS) for the network can be met. To do so, special congestion control bits have been incorporated into the address field of the Frame Relay: FECN and BECN. The basic idea is to avoid data accumulation inside the network.

FECN means Forward Explicit Congestion Notification. The FECN bit can be set to 1 to indicate that congestion was experienced in the direction of the frame transmission, so it informs the **destination** that congestion has occurred. BECN means Backwards Explicit Congestion Notification. The BECN bit can be set to 1 to indicate that congestion was experienced in the network in the direction opposite of the frame transmission, so it informs the *sender* that congestion has occurred.

Frame Relay versus X.25

X.25 provides quality of service and error-free delivery, whereas, Frame Relay was designed to relay data as quickly as possible over low error networks. Frame Relay eliminates a number of the higher-level procedures and fields used in X.25. Frame Relay was designed for use on links with error-rates far lower than available when X.25 was designed.

X.25 prepares and sends packets, while Frame Relay prepares and sends frames. X.25 packets contain several fields used for error checking and flow control, most of which are not used by Frame Relay. The frames in Frame Relay contain an expanded link layer address field that enables Frame Relay nodes to direct frames to their destinations with minimal processing. The elimination of functions and fields over X.25 allows Frame Relay to move data more quickly, but leaves more room for errors and larger delays should data need to be retransmitted.

X.25 packet switched networks typically allocated a fixed bandwidth through the network for each X.25 access, regardless of the current load. This resource allocation approach, while apt for applications that require guaranteed quality of service, is inefficient for applications that are highly dynamic in their load characteristics or which would benefit from a more dynamic resource allocation. Frame Relay networks can dynamically allocate bandwidth at both the physical and logical channel level.

Virtual circuits

As a WAN protocol, Frame Relay is most commonly implemented at Layer 2 (data link layer) of the Open Systems Interconnection (OSI) seven layer model. Two types of circuits exist: permanent virtual circuits (PVCs) which are used to form logical end-to-end links mapped over a physical network, and switched virtual circuits (SVCs). The latter are analogous to the circuit-switching concepts of the public switched telephone network (PSTN), the global phone network.

Frame Relay origins

Frame Relay began as a stripped-down version of the X.25 protocol, releasing itself from the error-correcting burden most commonly associated with X.25. When Frame Relay detects an error, it simply drops the offending packet. Frame Relay uses the concept of shared-access and relies on a technique referred to as "best-effort", whereby error-correction practically does not exist and practically no guarantee of reliable data delivery occurs. Frame Relay provides an industry-standard encapsulation utilizing the strengths of high-speed, packet-switched technology able to service multiple virtual circuits and protocols between connected devices, such as two routers.

Local Management Interface (LMI)

Initial proposals for Frame Relay were presented to the Consultative Committee on International Telephone and Telegraph (CCITT) in 1984. Lack of interoperability and standardization, prevented any significant Frame Relay deployment until 1990 when Cisco, Digital Equipment Corporation (DEC), Northern Telecom, and StrataCom formed a consortium to focus on its development. They produced a protocol that provided additional capabilities for complex inter-networking environments. These Frame Relay extensions are referred to as the Local Management Interface (LMI).

Datalink connection identifiers (DLCIs) are numbers that refer to paths through the Frame Relay network. They are only locally significant, which means that when device-A sends data to device-B it will most-likely use a different DLCI than device-B would use to reply. Multiple virtual circuits can be active on the same physical end-points (performed by using subinterfaces).

The LMI global addressing extension gives Frame Relay data-link connection identifier (DLCI) values global rather than local significance. DLCI values become DTE addresses that are unique in the Frame Relay WAN. The global addressing extension adds functionality and manageability to Frame Relay internetworks. Individual network interfaces and the end nodes attached to them, for example, can be identified by using standard address-resolution and discovery techniques. In addition, the entire Frame Relay network appears to be a typical LAN to routers on its periphery.

LMI virtual circuit status messages provide communication and synchronization between Frame Relay DTE and DCE devices. These messages are used to periodically report on the status of PVCs, which prevents data from being sent into black holes (that is, over PVCs that no longer exist).

The LMI multicasting extension allows multicast groups to be assigned. Multicasting saves bandwidth by allowing routing updates and address-resolution messages to be sent only to specific groups of routers. The extension also transmits reports on the status of multicast groups in update messages.

Committed information rate (CIR)

Frame Relay connections are often given a committed information rate (CIR) and an allowance of burstable bandwidth known as the extended information rate (EIR). The provider guarantees that the connection will always support the CIR rate, and sometimes the EIR rate should there be adequate bandwidth. Frames that are sent in excess of the CIR are marked as *discard eligible* (DE) which means they can be dropped should congestion occur within the Frame Relay network. Frames sent in excess of the EIR are dropped immediately. All traffic exceeding the CIR is marked discard eligible.

Market reputation

Frame Relay aimed to make more efficient use of existing physical resources, permitting the over-provisioning of data services by telecommunications companies to their customers, as clients were unlikely to be utilizing a data service 100 percent of the time. In more recent years, Frame Relay has acquired a bad reputation in some markets because of excessive bandwidth overbooking.

Telcos often sell Frame Relay to businesses looking for a cheaper alternative to dedicated lines; its use in different geographic areas depended greatly on governmental and telecommunication companies' policies. Some of the early companies to make Frame Relay products included StrataCom (later acquired by Cisco Systems) and Cascade Communications (later acquired by Ascend Communications and then by Lucent Technologies).

As of June 2007 AT&T was the largest Frame Relay service provider in the USA, with local networks in 22 states, plus national and international networks.

References

External links

- RFC 1490 - Multiprotocol Interconnect over Frame Relay
- RFC 1973 - PPP in Frame Relay
- RFC 2427 - Multiprotocol Interconnect over Frame Relay
- The IP/MPLS Forum (MPLS, Frame Relay, and ATM) (http://www.ipmplsforum.org/)
- Cisco Frame Relay Tutorial (http://docwiki.cisco.com/wiki/Frame_Relay)
- Frame Relay animation (http://www.visualland.net/view.php?cid=1336&protocol=Frame Relay&title=1. Frame Relay Basics&ctype=2)

Frame synchronization

While receiving a stream of framed data, **frame synchronization** is the process by which incoming frame alignment signals, *i.e.*, distinctive bit sequences (a syncword), are identified, *i.e.*, distinguished from data bits, permitting the data bits within the frame to be extracted for decoding or retransmission. This is sometimes referred to as "framing".

Framing

Frame Synchronized PCM Stream -- Telemetry Application

If the transmission is temporarily interrupted, or a bit slip event occurs, the receiver must re-synchronize.

The transmitter and the receiver must agree ahead of time on which frame synchronization scheme they will use. The most common frame synchronization schemes are:

- **Framing bit**. A common practice in telecommunications, for example in T-carrier, is to insert, in a dedicated time slot within the frame, a noninformation bit or **framing bit** that is used for synchronization of the incoming data with the receiver. In a bit stream, framing bits indicate the beginning or end of a frame. They occur at specified positions in the frame, do not carry information, and are usually repetitive.
- **Syncword framing**. Some systems use a special syncword at the beginning of every frame.
- **CRC-based framing**. Some telecommunications hardware uses CRC-based framing.

Frame synchronizer

Television

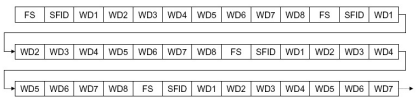
Further information: Time base corrector

A **frame synchronizer** is a device used in live television production to match the timing of an incoming video source to the timing of an existing video system. They are often used to "time in" consumer video equipment to a professional system but can be used to stabilize any video. The frame synchronizer essentially takes a picture of each frame of incoming video and then immediately outputs it with the correct synchronization signals to match an existing video system. A genlock signal is required to provide a means for video synchronizing with the house

reference.

Telemetry

In telemetry applications, a **frame synchronizer** is used to frame align a serial PCM (Pulse Code Modulated)binary stream.

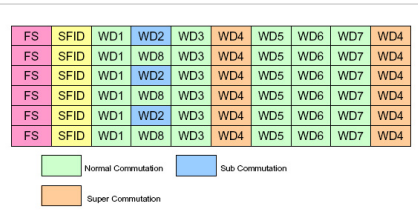


PCM Stream prior to Frame Synchronization

The Frame Synchronizer immediately follows the Bit Synchronizer in most telemetry applications. Without frame synchronization decommutation is impossible.

Frame Synchronized PCM Stream

The Frame Sync Pattern is a known binary pattern which repeats at a regular interval within the PCM stream. The frame synchronizer recognizes this pattern and aligns the data into Minor Frames or Sub Frames. Typically the frame sync pattern is followed by a counter (Sub-Frame ID) which dictates which minor or sub frame in the series is being transmitted. This becomes increasingly important in the decommutation stage where all data is deciphered as to what attribute was sampled. Different commutations require a constant awareness of which section of the major frame is being decoded.



Different types of commutation within a frame synchronized PCM stream

References

This article incorporates public domain material from the General Services Administration document "Federal Standard 1037C" [1] (in support of MIL-STD-188).

Scientific Articles

J. L. Massey. "Optimum frame synchronization ". IEEE trans. comm., com-20(2):115-119, April 1972.

R Scholtz. "Frame synchronization techniques", IEEE Transactions on Communications, 1980

P. Robertson. "Optimal Frame Synchronization for Continuous and Packet Data Transmission", PhD Dissertation, 1995, Fortschrittberichte VDI Reihe 10, Nr. 376 PDF [2]

External links

- Telemetry Ground Station Processing Electronics telemetry link [3]
- Algolith Frame Synchronizer Solutions [4]

References

[1] http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm
[2] http://www.kn-s.dlr.de/People/Robertson/Papers/diss.pdf
[3] http://www.ulyssix.com
[4] http://www.algolith.com/products/broadcast/frame-synchronization/index.html

Gateway (telecommunications)

Juniper SRX210 service gateway

In telecommunications, the term **gateway** has the following meaning:

- In a communications network, a network node equipped for interfacing with another network that uses different protocols.
  - A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
  - A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.
- Loosely, a computer or computer program configured to perform the tasks of a gateway. For a specific case, see default gateway.

Gateways, also called protocol converters, can operate at any network layer. The activities of a gateway are more complex than that of the router or switch as it communicates using more than one protocol.

Details

A gateway is a network point that acts as an entrance to another network. On the Internet, a node or stopping point node or a host (end-point) node. Both the computers of Internet users and the computers that serve pages to users are host nodes, while the nodes that connect the networks in between are gateways. For example, the computers that control traffic between company networks or the computers used by internet service providers (ISPs) to connect users to the internet are gateway nodes.

In the network for an enterprise, a computer server acting as a gateway node is often also acting as a proxy server and a firewall server. A gateway is often associated with both a router, which knows where to direct a given packet of data that arrives at the gateway, and a switch, which furnishes the actual path in and out of the gateway for a given packet.

On an IP network, clients should automatically send IP packets with a destination outside a given subnet mask to a network gateway. A subnet mask defines the IP range of a private network. For example, if a private network has a base IP address of 192.168.0.0 and has a subnet mask of 255.255.255.0, then any data going to an IP address outside of 192.168.0.X will be sent to that network's gateway. While forwarding an IP packet to another network, the gateway might or might not perform Network Address Translation.

A gateway is an essential feature of most routers, although other devices (such as any PC or server) can function as a gateway.

Most computer operating systems use the terms described above. A computer running Microsoft Windows however describes this standard networking feature as Internet Connection Sharing; which will act as a gateway, offering a connection between the Internet and an internal network. Such a system might also act as a DHCP server. Dynamic Host Configuration Protocol (DHCP) is a protocol used by networked devices (clients) to obtain various parameters necessary for the clients to operate in an Internet Protocol (IP) network. By using this protocol, system administration workload greatly decreases, and devices can be added to the network with minimal or no manual configurations.

Internet-to-Orbit Gateway

An Internet to orbit gateway (I2O) is a machine that acts as a connector between computers or devices connected to the Internet and computer systems orbiting the earth, like satellites or even manned spacecrafts. Such connection is made when the I2O establishes a stable link between the spacecraft and a computer or a network of computers on the Internet, such link can be control signals, audio frequency, or even visible spectrum signals.

Project HERMES is the first project to have brought this kind of machine into operation. The HERMES-A/MINOTAUR Space Flight Control Center[1] became operative on June 6, 2009 and was operated by representatives of 34 countries on the UNOOSA Symposium of Small Satellites for Sustainable Development[2] in Graz, Austria on September 10, 2009. Project HERMES is an initiative of the Ecuadorian Civilian Space Agency and has a maximum coverage of 22,000 km, HERMES-A is supposed to be the first gateway of a network of five covering all South America. HERMES-A/MINOTAUR is not only capable of data transmission but voice also.

Project GENSO is an initiative from NASA and ESA, and it is expected to begin operations on April 2010, it is supposed to have worldwide coverage.

Cloud Gateway

A Cloud storage gateway is a network appliance or server which resides at the customer premises and translates cloud storage APIs such as SOAP or REST to block-based storage protocols such as iSCSI or Fibre Channel or file-based interfaces such as NFS or CIFS.[3] [4] Cloud storage gateways enable companies to integrate cloud storage into applications without moving the applications into the cloud.[5] In addition they simplify data protection.[5]

External Links

- PortFusion [6] an open-source, tiny, multi-protocol, distributed reverse proxy for gateway servers developed at the University of Heidelberg.

Sources

- Federal Standard 1037C
- MIL-STD-188

References

[1] http://exa.ec/bp25
[2] http://www.unoosa.org/pdf/sap/2009/graz/Programme_3September.pdf
[3] GigaOm, Show Me the Gateway — Taking Storage to the Cloud, By Gary Orenstein. 22 June 2010 (http://gigaom.com/2010/06/22/show-me-the-gateway-taking-storage-to-the-cloud/)
[4] Boles, Jeff. Cloud file storage pros and cons (http://searchstorage.techtarget.com/generic/0,295582,sid5_gci1528586,00.html). 15 March 2011
[5] Slack, Eric. Hybrid Cloud Appliances Simplify Data Storage Infrastructure (http://www.storage-switzerland.com/Articles/Entries/2011/6/14_Hybrid_Cloud_Appliances_Simplify_Data_Storage_Infrastructure.html). June 14, 2011
[6] http://sourceforge.net/p/portfusion/

Internet protocol suite

The **Internet protocol suite** is the set of communications protocols used for the Internet and other similar networks. It is commonly known as **TCP/IP** from its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP), which were the first networking protocols defined in this standard. Modern IP networking represents a synthesis of several developments that began to evolve in the 1960s and 1970s, namely the precursors of the Internet and local area networks, which emerged during the 1980s, together with the advent of the World Wide Web in the early 1990s.

The Internet protocol suite classifies its methods and protocols into four hierarchical abstraction layers. From the lowest to the highest communication layer, these are the link layer, the internet layer, the transport layer, and the application layer.[1] [2] The layers define the operational scope or reach of the protocols in each layer, reflected loosely in the layer names. Each layer has functionality that solves a set of problems in its scope.

The link layer contains communication technologies for the local network to which the host is connected directly by hardware components. The internet layer facilitates the interconnection of local networks. As such, this layer establishes the Internet. Host-to-host communication tasks are handled in the transport layer, which provides a general application-agnostic framework to transmit data between hosts using protocols like the Transmission Control Protocol and the User Datagram Protocol (UDP). Finally, the highest-level application layer contains all protocols that are defined each specifically for the functioning of the vast array of data communications services. This layer handles application-based interaction on a process-to-process level between communicating Internet hosts.

History

The Internet protocol suite resulted from research and development conducted by the Defense Advanced Research Projects Agency (DARPA) in the early 1970s. After initiating the pioneering ARPANET in 1969, DARPA started work on a number of other data transmission technologies. In 1972, Robert E. Kahn joined the DARPA Information Processing Technology Office, where he worked on both satellite packet networks and ground-based radio packet networks, and recognized the value of being able to communicate across both. In the spring of 1973, Vinton Cerf, the developer of the existing ARPANET Network Control Program (NCP) protocol, joined Kahn to work on open-architecture interconnection models with the goal of designing the next protocol generation for the ARPANET.

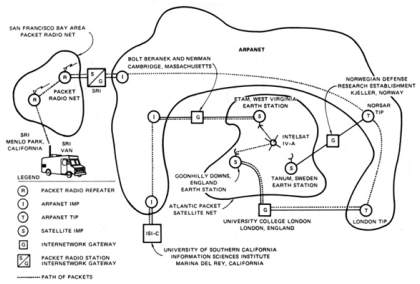


Diagram of the first internetworked connection

A Stanford Research Institute packet radio van, site of the first three-way internetworked transmission.

By the summer of 1973, Kahn and Cerf had worked out a fundamental reformulation, where the differences between network protocols were hidden by using a common internetwork protocol, and, instead of the network being responsible for reliability, as in the ARPANET, the hosts became responsible. Cerf credits Hubert Zimmerman and Louis Pouzin, designer of the CYCLADES network, with important influences on this design.

The network's design included the recognition it should provide only the functions of efficiently transmitting and routing traffic between end nodes and that all other intelligence should be located at the edge of the network, in the end nodes. Using a simple

design, it became possible to connect almost any network to the ARPANET, irrespective of their local characteristics, thereby solving Kahn's initial problem. One popular expression is that TCP/IP, the eventual product of Cerf and Kahn's work, will run over "*two tin cans and a string*."

A computer, called a router, is provided with an interface to each network. It forwards packets back and forth between them.[3] Originally a router was called *gateway*, but the term was changed to avoid confusion with other types of gateways.

The idea was worked out in more detailed form by Cerf's networking research group at Stanford in the 1973–74 period, resulting in the first TCP specification.[4] The early networking work at Xerox PARC, which produced the PARC Universal Packet protocol suite, much of which existed around the same period of time, was also a significant technical influence.

DARPA then contracted with BBN Technologies, Stanford University, and the University College London to develop operational versions of the protocol on different hardware platforms. Four versions were developed: TCP v1, TCP v2, a split into TCP v3 and IP v3 in the spring of 1978, and then stability with TCP/IP v4 — the standard protocol still in use on the Internet today.

In 1975, a two-network TCP/IP communications test was performed between Stanford and University College London (UCL). In November, 1977, a three-network TCP/IP test was conducted between sites in the US, UK, and Norway. Several other TCP/IP prototypes were developed at multiple research centers between 1978 and 1983. The migration of the ARPANET to TCP/IP was officially completed on flag day January 1, 1983, when the new protocols were permanently activated.[5]

In March 1982, the US Department of Defense declared TCP/IP as the standard for all military computer networking.[6] In 1985, the Internet Architecture Board held a three day workshop on TCP/IP for the computer industry, attended by 250 vendor representatives, promoting the protocol and leading to its increasing commercial use.

In 1985 the first Interop conference was held, focused on network interoperability via further adoption of TCP/IP. It was founded by Dan Lynch, one of early Internet activists. It was attended from the beginning by large corporations such as IBM and DEC. Interoperability conferences have been held since then every year, and every year from 1985 through 1993 the number of attendees tripled. [citation needed]

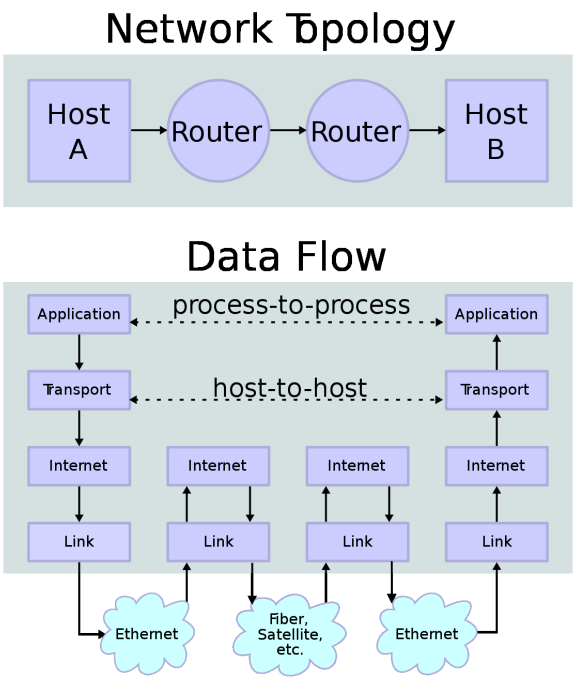
IBM, ATT and DEC were the first major corporations to adopt TCP/IP despite having competing internal protocols (SNA, XNS, etc) and politics. In IBM the TCP/IP development was undertaken from 1984 onward in the group of Barry Appelman who later moved to AOL to be the head of all AOL's development efforts. At IBM Barry Appelman with a handful of developers was able to maneuver the corporate politics to get a stream of TCP/IP products for various IBM systems - MVS, VM, OS/2 among others. At the same time several smaller companies began offering TCP/IP stacks for DOS and MS Windows, such as FTP Software, Wollongong [7]. The first VM/CMS TCP/IP stack came from university of Wisconsin[8] . Back then most of these TCP/IP stacks were written single-handedly by very few talented programmers. For example, John Romkey of FTP Software was the author of the MIT PC/IP package as well as at a later point of FTP Software[9] . John Romkey's PC/IP implementation was the first IBM PC TCP/IP stack. Jay Elinsky and Oleg Vishnepolsky of IBM Research wrote VM/CMS and OS/2 TCP/IP stacks, respectively[10] .

The spread of TCP/IP was fueled further in June 1989 when AT&T agreed to put the TCP/IP code developed for UNIX system at University of Berkeley into public domain. Known as BSD, this code base was adopted by a variety of vendors, IBM included, for their own TCP/IP stacks. Until Microsoft released TCP/IP stack in 1995 with its Windows 95, many companies marketed and sold TCP/IP software for Windows. Windows 95 support for TCP/IP came a little late in the Internet evolution, but when it finally came, the dominance of TCP/IP over other protocols became insurmountable. IBM's SNA, OSI, Microsoft's native NetBIOS, Xerox' XNS and all other protocols disappeared from the commercial landscape. [citations needed]

Layers in the Internet protocol suite

The concept of layers

The Internet protocol suite uses encapsulation to provide abstraction of protocols and services. Encapsulation is usually aligned with the division of the protocol suite into layers of general functionality. In general, an application (the highest level of the model) uses a set of protocols to send its data down the layers, being further encapsulated at each level.



Instantiations of the TCP/IP stack operating on two hosts each connected to its router on the Internet. The flow of user data through the layers used at each hop is shown.

According to RFC 1122, the Internet protocol suite organizes the functional groups of protocols and methods into four layers, the application layer, the transport layer, the internet layer, and the link layer. This model was not intended to be a rigid reference model into which new protocols have to fit in order to be accepted as a standard.

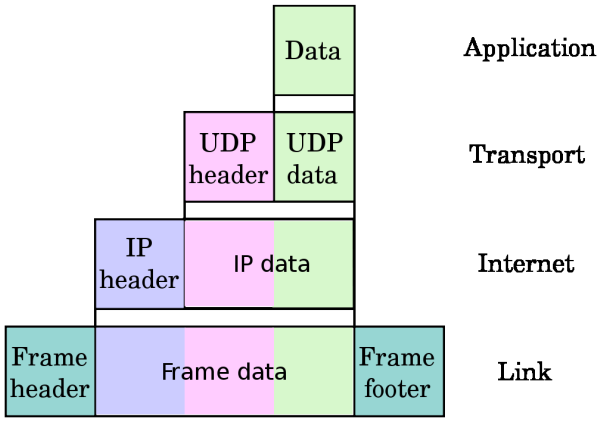
The role of layering in TCP/IP may be illustrated by an example network scenario (right-hand diagram), in which two Internet host computers communicate across local network boundaries constituted by their internetworking routers. The application on each host executes read and write operations as if the processes were directly connected to each other by some kind of data pipe, every other detail of the communication is hidden from each process. The underlying mechanisms that transmit data between the host computers are located in the lower protocol layers.

The transport layer establishes host-to-host connectivity, meaning it handles the details of data transmission that are independent of the structure of user data and the logistics of exchanging information for any particular specific purpose. The layer simply establishes a basic data channel that an application uses in its task-specific data exchange. For this purpose the layer establishes the concept of the *port*, a numbered logical construct allocated specifically for each of the communication channels an application needs. For many types of services, these port numbers have been standardized so that client computers may address specific services of a server computer without the involvement of service announcements or directory services.

The transport layer operates on top of the internet layer. The internet layer is not only agnostic of application data structures as the transport layer, but it also does not distinguish between operation of the various transport layer protocols. It only provides an unreliable datagram transmission facility between hosts located on potentially different IP networks by forwarding the transport layer datagrams to an appropriate next-hop router for further relaying to its destination. With this functionality, the internet layer makes possible internetworking, the interworking of different IP networks, and it essentially establishes the Internet. The Internet Protocol is the principal component of the internet layer, and it defines two addressing systems to identify network hosts computers, and to locate them on the network. The original address system of the ARPANET and its successor, the Internet, is Internet Protocol version 4 (IPv4). It uses a 32-bit IP address and is therefore capable of identifying approximately four billion hosts. This limitation was eliminated by the standardization of Internet Protocol version 6 (IPv6) in 1998, and beginning production implementations in approximately 2006.

The lowest layer in the Internet protocol suite is the link layer. It comprises the tasks of specific networking requirements on the local link, the network segment that a hosts network interface is connected to. This involves interacting with the hardware-specific functions of network interfaces and specific transmission technologies.

As the user data, first manipulated and structured in the application layer, is passed through the descending layers of the protocol stack each layer adds encapsulation information as illustrated in the diagram (right). A receiving host reverses the encapsulation at each layer by extracting the higher level data and passing it up the stack to the receiving process.



Successive encapsulation of application data descending through the protocol stack before transmission on the local network link

Layer names and number of layers in the literature

The following table shows the layer names and the number of layers of networking models presented in RFCs and textbooks in widespread use in today's university computer networking courses.

RFC 1122 [11]	Tanenbaum	Cisco Academy[12]	Kurose[13] Forouzan [14]	Comer[15] Kozierok[16]	Stallings[17]	Arpanet Reference Model 1982 (RFC 871)
Four layers [18]	*Four layers* [19]	*Four layers*	*Five layers*	*Four+one layers*	*Five layers*	*Three layers*
"Internet model"	"TCP/IP reference model"[20]	"Internet model"	"Five-layer Internet model" or "TCP/IP protocol suite"	"TCP/IP 5-layer reference model"	"TCP/IP model"	"Arpanet reference model"
Application [18] [21]	Application	Application	Application	Application	Application	Application/Process
Transport [18]	Transport	Transport	Transport	Transport	Host-to-host or transport	Host-to-host
Internet [18]	Internet	Internetwork	Network	Internet	Internet	
Link [18]	Host-to-network	Network interface	Data link	Data link (Network interface)	Network access	Network interface
			Physical	(Hardware)	Physical	

These textbooks are secondary sources that may contravene the intent of RFC 1122 and other IETF primary sources.[22]

Different authors have interpreted the RFCs differently regarding the question whether the link layer (and the TCP/IP model) covers OSI model layer 1 (physical layer) issues, or if a hardware layer is assumed below the link layer. Some authors have tried to use other names for the link layer, such as *network interface layer*, in view to avoid confusion with the similarly named OSI model layer 2 (data link layer) of the seven layer OSI model. Others have attempted to map the Internet Protocol model onto the OSI model. The mapping often results in a model with five layers where the link layer is split into an OSI model layer 2 (data link layer) on top of an OSI model layer 1 (physical layer). In literature with a bottom-up approach to Internet communication,[14] [15] [17] in which hardware issues are emphasized, those are often discussed in terms of OSI model layer 1 (physical layer) and OSI model layer 2 (data link layer).

The internet layer is usually directly mapped into OSI model layer 3 (network layer), a more general concept of network functionality. The transport layer of the TCP/IP model, sometimes also described as the host-to-host layer, is mapped to OSI layer 4 (transport layer), sometimes also including aspects of OSI layer 5 (session layer) functionality. OSI layer 7 (application layer), OSI layer 6 (presentation layer), and the remaining functionality of OSI layer 5 (session layer) are collapsed into TCP/IP's application layer. The argument is that these OSI layers do not usually exist as separate processes and protocols in Internet applications.

However, the Internet protocol stack has never been altered by the Internet Engineering Task Force from the four layers defined in RFC 1122. The IETF makes no effort to follow the OSI model although RFCs sometimes refer to it. The IETF has repeatedly stated that Internet protocol and architecture development is not intended to be compliant to the OSI model.

RFC 3439, addressing Internet architecture, contains a section entitled: "Layering Considered Harmful".[22]

Implementations

Most computer operating systems in use today, including all consumer-targeted systems, include a TCP/IP implementation.

Minimally acceptable implementation includes implementation for (from most essential to the less essential) IP, ARP, ICMP, UDP, TCP and sometimes IGMP. It is in principle possible to support only one of transport protocols (i.e. simple UDP), but it is rarely done, as it limits usage of the whole implementation. IPv6, beyond own version of ARP (NBP), and ICMP (ICMPv6), and IGMP (IGMPv6) have some additional required functions, and often is accompanied with integrated IPSec security layer. Other protocols could be easily added later (often they can be implemented entirely in the userspace), for example DNS for resolving domain names to IP addresses or DHCP client for automatic configuration of network interfaces.

Most of the IP implementations are accessible to the programmers using socket abstraction (usable also with other protocols) and proper API for most of the operations. This interface is known as BSD sockets and was used initially in C.

Unique implementations include Lightweight TCP/IP, an open source stack designed for embedded systems and KA9Q NOS, a stack and associated protocols for amateur packet radio systems and personal computers connected via serial lines.

References

[1] RFC 1122, *Requirements for Internet Hosts – Communication Layers*, R. Braden (ed.), October 1989
[2] RFC 1123, *Requirements for Internet Hosts – Application and Support*, R. Braden (ed.), October 1989
[3] RFC 1812, *Requirements for IP Version 4 Routers*, F. Baker (June 1995)
[4] RFC 675, *Specification of Internet Transmission Control Protocol*, V. Cerf et al. (December 1974)
[5] Internet History (http://www.livinginternet.com/i/ii.htm)
[6] Ronda Hauben. "From the ARPANET to the Internet" (http://www.columbia.edu/~rh120/other/tcpdigest_paper.txt). TCP Digest (UUCP). . Retrieved 2007-07-05.
[7] http://support.microsoft.com/kb/108007
[8] (http://www.weblab.isti.cnr.it/education/ssfs/lezioni/slides/archives/cern.htm)
[9] (http://www.romkey.com/about/)
[10] Barry Appelman
[11] IETF, RFC 1122, p.7, "To communicate using the Internet system, a host must implement the layered set of protocols comprising the Internet protocol suite. A host typically must implement at least one protocol from each layer."
[12] Mark Dye, Mark A. Dye, Wendell, Network Fundamentals: CCNA Exploration Companion Guide, 2007, ISBN 1-58713-208-7
[13] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach, 2008, ISBN 0-321-49770-8 (http://www.pearsonhighered.com/educator/academic/product/0,,0321497708,00+en-USS_01DBC.html)
[14] Behrouz A. Forouzan, Data Communications and Networking (http://books.google.com/books?id=U3Gcf65Pu9IC&printsec=frontcover&dq=forouzan+"computer+networks"#PPA29,M1)

[15] Douglas E. Comer, Internetworking with TCP/IP: Principles, Protocols and Architecture, Pearson Prentice Hall 2005, ISBN 0-13-187671-6 (http://books.google.com/books?id=jonyuTASbWAC&pg=PA155&cad=0_0#PPA161,M1)
[16] Charles M. Kozierok, "The TCP/IP Guide", No Starch Press 2005 (http://books.google.com/books?id=Pm4RgYV2w4YC&pg=PA131&dq="TCP/IP+model+layers"#PPA129,M1)
[17] William Stallings, Data and Computer Communications, Prentice Hall 2006, ISBN 0-13-243310-9 (http://books.google.com/books?id=c_AWmhkovR0C&pg=PA35&dq="internet+layer"+"network+access+layer")
[18] IETF, RFC 1122, p.7-8, "The protocol layers [...] are as follows [...]:[...] Application Layer [...] Transport Layer [...] Internet Layer [...] Link Layer"
[19] Andrew Tanenbaum, Computer Networks, section 1.4.3, "[...] the OSI model has seven layers and the TCP/IP has four layers."
[20] Tanenbaum, Andrew S. (2002). *Computer Networks*. Prentice Hall. p. 41. ISBN 0130661023. "1.4.2 The TCP/IP Reference Model" Excerpt at Google Books (http://books.google.com/books?id=Pd-z64SJRBAC&pg=PA42&vq=internet+layer&dq=networks#PPA42,M1)
[21] IETF, RFC 1122, p.8, "The application layer is the top layer of the Internet protocol suite."
[22] R. Bush; D. Meyer (December 2002). *Some Internet Architectural Guidelines and Philosophy* (http://www.isi.edu/in-notes/rfc3439.txt). Internet Engineering Task Force. . Retrieved 2007-11-20

Further reading

- Douglas E. Comer. *Internetworking with TCP/IP - Principles, Protocols and Architecture*. ISBN 86-7991-142-9
- Joseph G. Davies and Thomas F. Lee. *Microsoft Windows Server 2003 TCP/IP Protocols and Services*. ISBN 0-7356-1291-9
- Forouzan, Behrouz A. (2003). *TCP/IP Protocol Suite* (2nd ed.). McGraw-Hill. ISBN 0-07-246060-1.
- Craig Hunt *TCP/IP Network Administration*. O'Reilly (1998) ISBN 1-56592-322-7
- Maufer, Thomas A. (1999). *IP Fundamentals*. Prentice Hall. ISBN 0-13-975483-0.
- Ian McLean. *Windows(R) 2000 TCP/IP Black Book*. ISBN 1-57610-687-X
- Ajit Mungale *Pro .NET 1.1 Network Programming*. ISBN 1-59059-345-6
- W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. ISBN 0-201-63346-9
- W. Richard Stevens and Gary R. Wright. *TCP/IP Illustrated, Volume 2: The Implementation*. ISBN 0-201-63354-X
- W. Richard Stevens. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*. ISBN 0-201-63495-3
- Andrew S. Tanenbaum. *Computer Networks*. ISBN 0-13-066102-3
- Clark, D. (1988). "The Design Philosophy of the DARPA Internet Protocols" (http://www.cs.princeton.edu/~jrex/teaching/spring2005/reading/clark88.pdf). *SIGCOMM '88 Symposium proceedings on Communications architectures and protocols* (ACM): 106–114. doi:10.1145/52324.52336. Retrieved 2011-10-16.

External links

- Internet History (http://www.livinginternet.com/i/ii.htm) -- Pages on Robert Kahn, Vinton Cerf, and TCP/IP (reviewed by Cerf and Kahn).
- RFC 675 (http://www.ietf.org/rfc/rfc0675.txt) - Specification of Internet Transmission Control Program, December 1974 Version
- TCP/IP State Transition Diagram (http://www.night-ray.com/TCPIP_State_Transition_Diagram.pdf) (PDF)
- RFC 1180 A TCP/IP Tutorial - from the Internet Engineering Task Force (January 1991)
- TCP/IP FAQ (http://www.itprc.com/tcpipfaq/)
- TCP/IP Resources List (http://www.private.org.il/tcpip_rl.html)
- The TCP/IP Guide (http://www.tcpipguide.com/free/) - A comprehensive look at the protocols and the procedures/processes involved
- A Study of the ARPANET TCP/IP Digest (http://www.columbia.edu/~rh120/other/tcpdigest_paper.txt)
- TCP/IP Sequence Diagrams (http://www.eventhelix.com/RealtimeMantra/Networking/)
- The Internet in Practice (http://www.searchandgo.com/articles/internet/internet-practice-4.php)
- TCP/IP - Directory & Informational Resource (http://softtechinfo.com/network/tcpip.html)

- Daryl's TCP/IP Primer (http://www.ipprimer.com) - Intro to TCP/IP LAN administration, conversational style
- Introduction to TCP/IP (http://www.linux-tutorial.info/MContent-142)
- TCP/IP commands from command prompt (http://blog.webgk.com/2007/10/dns-tcpip-commands-from-command-prompt.html)
- cIPS (http://sourceforge.net/projects/cipsuite/) — Robust TCP/IP stack for embedded devices without an Operating System

Label switching

Label switching is a technique of network relaying to overcome the problems perceived by traditional IP-table switching (also known as traditional layer 3 hop-by-hop routing[1]). Here, the switching of network packets occurs at a lower level, namely the data link layer rather than the traditional network layer.

Each packet is assigned a label number and the switching takes place after examination of the label assigned to each packet. The switching is much faster than IP-routing. New technologies such as Multiprotocol Label Switching (MPLS) use label switching. The established ATM protocol also uses label switching at its core.

According to RFC 2475 (An Architecture for Differentiated Services, December 1998): "Examples of the label switching (or virtual circuit) model include Frame Relay, ATM, and MPLS. In this model path forwarding state and traffic management or Quality of Service (QoS) state is established for traffic streams on each hop along a network path. Traffic aggregates of varying granularity are associated with a label switched path at an ingress node, and packets/cells within each label switched path are marked with a forwarding label that is used to look up the next-hop node, the per-hop forwarding behavior, and the replacement label at each hop. This model permits finer granularity resource allocation to traffic streams, since label values are not globally significant but are only significant on a single link; therefore resources can be reserved for the aggregate of packets/cells received on a link with a particular label, and the label switching semantics govern the next-hop selection, allowing a traffic stream to follow a specially engineered path through the network."
A related topic is "Multilayer Switching," which discusses silicon-based wire-speed routing devices that examine not only layer 3 packet information, but also layer 4 (transport) and layer 7 (application) information.

References

[1] Label Switching (Linktionary term) (http://www.linktionary.com/l/label_switching.html)

- linktionary.com (http://www.linktionary.com/l/label_switching.html)
- (http://www.linktionary.com/l/label_switching.html), “Label Switching”, copyright 2001, by Tom Sheldon and Big Sur Multimedia, accessed on June 29, 2008.

LAN switching

This article addresses packet switching in computer networks.

LAN switching is a form of packet switching used in local area networks. Switching technologies are crucial to network design, as they allow traffic to be sent only where it is needed in most cases, using fast, hardware-based methods.

Layer 2 switching

Layer 2 switching is hardware based,[1] which means it uses the media access control address (MAC address) from the host's network interface cards (NICs) to decide where to forward frames. Switches use application-specific integrated circuit (ASICs) to build and maintain filter tables (also known as MAC address tables). One way to think of a layer 2 switch is as a multiport bridge.

Layer 2 switching provides the following

- Hardware-based bridging (MAC)
- Wire speed
- High speed
- Low latency
- Low cost

Layer 2 switching is highly efficient because there is no modification to the data packet, only to the frame encapsulation of the packet, and only when the data packet is passing through dissimilar media (such as from Ethernet to FDDI). Layer 2 switching is used for workgroup connectivity and network segmentation (breaking up collision domains). This allows a flatter network design with more network segments than traditional 10BaseT shared networks. Layer 2 switching has helped develop new components in the network infrastructure

- Server farms — Servers are no longer distributed to physical locations because virtual LANs can be created to create broadcast domains in a switched internetwork. This means that all servers can be placed in a central location, yet a certain server can still be part of a workgroup in a remote branch, for example.
- Intranets — Allows organization-wide client/server communications based on a Web technology.

These new technologies allow more data to flow off from local subnets and onto a routed network, where a router's performance can become the bottleneck.

Limitations

Layer 2 switches have the same limitations as bridge networks. Bridges are good if a network is designed by the 80/20 rule: users spend 80 percent of their time on their local segment.

Bridged networks break up collision domains, but the network remains one large broadcast domain. Similarly, layer 2 switches (bridges) cannot break up broadcast domains, which can cause performance issues and limits the size of your network. Broadcast and multicasts, along with the slow convergence of spanning tree, can cause major problems as the network grows. Because of these problems, layer 2 switches cannot completely replace routers in the internetwork.

Layer 3 switching

The only difference between a layer 3 switch and router is the way the administrator creates the physical implementation. Also, traditional routers use microprocessors to make forwarding decisions, and the switch performs only hardware-based packet switching. However, some traditional routers can have other hardware functions as well in some of the higher-end models. Layer 3 switches can be placed anywhere in the network because they handle high-performance LAN traffic and can cost-effectively replace routers. Layer 3 switching is all hardware-based packet forwarding, and all packet forwarding is handled by hardware ASICs. Layer 3 switches really are no different functionally than a traditional router and perform the same functions, which are listed here

- Determine paths based on logical addressing
- Run layer 3 checksums (on header only)
- Use Time to Live (TTL)
- Process and respond to any option information
- Update Simple Network Management Protocol (SNMP) managers with Management Information Base (MIB) information
- Provide Security

The benefits of layer 3 switching include the following

- Hardware-based packet forwarding
- High-performance packet switching
- High-speed scalability
- Low latency
- Lower per-port cost
- Flow accounting
- Security
- Quality of service (QoS)

1. SWITCHING:

Switching algorithm is relatively simple. it is the same for most of the routing protocols. in most cases a host determines that it must send a packet to another host. Having acquired a routers address by some means, the source host send a packet address specifically to a routers physical(MAC_) address, this time with the protocol(network layer)address of the destination host.

As it examines the packets destinations protocole address, the router determines that it either knows or does not know how to forward packet to the next hop. If the router does not know how to do it, it typically drops the packet. if it knows how to forward packet, however, it changes the destinations physical address to that of the next hop n transmittess the packet.

The next hop may be the destination or the next router, which executes the same switching process. As the packet moves through the internetwork, its physical address changes, but its protocol address remains same.

IEEE has developed the hierarchical terminology that is useful in describing this process. The network devices without capability to forward packets between subnetworks are called end system(ES), whereas network devices with this capabilities are called intermediate systems(IS). IS are further divided into those that can communicate within routing domain(Intradomain ES) and those that communicate both within and between routing domains(Interdomains IS) a routing domain is generally considered as portion of an internetwork under common administrative authority that is regulated by a particular set of administrative guidlines. Routing domains are also called as autonomous systems.

Layer 4 switching

Layer 4 switching is considered a hardware-based layer 3 switching technology that can also consider the application used (for example, Telnet or FTP).

Layer 4 switching provides additional routing above layer 3 by using the port numbers found in the Transport layer header to make routing decisions.

These port numbers are found in Request for Comments (RFC) 1700 and reference the upper-layer protocol, program, or application.

Layer 4 information has been used to help make routing decisions for quite a while. For example, extended access lists can filter packets based on layer 4 port numbers. Another example is accounting information gathered by open standards using sFlow provided by companies like Arista Networks or proprietary solutions like NetFlow switching in Cisco's higher-end routers.

The largest benefit of layer 4 switching is that the network administrator can configure a layer 4 switch to prioritize data traffic by application, which means a QoS can be defined for each user.

For example, a number of users can be defined as a Video group and be assigned more priority, or band-width, based on the need for video conferencing.

Multi-layer switching (MLS)

Multi-layer switching combines layer 2, 3, and 4 switching technologies and provides high-speed scalability with low latency. It accomplishes this high combination of high-speed scalability with low latency by using huge filter tables based on the criteria designed by the network administrator.

Multi-layer switching can move traffic at wire speed and also provide layer 3 routing, which can remove the bottleneck from the network routers. This technology is based on the idea of "route once, switch many".

Multi-layer switching can make routing/switching decisions based on the following

- MAC source/destination address in a Data Link frame
- IP source/destination address in the Network layer header
- Protocol field in the Network layer header
- Port source/destination numbers in the Transport layer header

There is no performance difference between a layer 3 and a layer 4 switch because the routing/switching is all hardware based.

References

[1] Techsoft Computing (http://www.techsoftcomputing.com/internetworkdesign/nd2002.html#wp3310)

External links

- Arista Networks (http://www.aristanetworks.com/) industries leading 10G data center switching products
- Cisco IT Case Studies (http://www.cisco.com/web/about/ciscoitatwork/network_systems/index.html) about routing and switching
- Is Switching a Commodity?" (http://www.youtube.com/watch?v=FjhrXwgirQc&feature=PlayList&p=B0C5AFFFD7762E3C&index=10) Fact or Fiction with Jennifer Geisler (Flash software required)

Link layer

In computer networking, the **link layer** is the lowest layer in the Internet Protocol Suite (commonly known as "TCP/IP"), the networking architecture of the Internet (RFC 1122, RFC 1123). It is the group of methods or protocols that only operate on a host's link. The link is the physical and logical network component used to interconnect hosts or nodes in the network and a link protocol is a suite of methods and standards that operate only between adjacent network nodes of a Local area network segment or a wide area network connection.

Despite the different semantics of layering in TCP/IP and OSI, the link layer is often described as a combination of the data link layer (layer 2) and the physical layer (layer 1) in the Open Systems Interconnection (OSI) protocol stack. However, TCP/IP's layers are *descriptions* of operating scopes (application, host-to-host, network, link) and not detailed *prescriptions* of operating procedures, data semantics, or networking technologies.

RFC 1122 exemplifies that local area network protocols such as Ethernet and IEEE 802, and framing protocols such as Point-to-Point Protocol (PPP) belong to the link layer.

Definition in standards and text books

LAN standards such as Ethernet and IEEE 802 specifications use terminology from the seven-layer OSI model rather than the TCP/IP reference model. The TCP/IP model in general does not consider physical specifications, rather it assumes a working network infrastructure that can deliver media level frames on the link. Therefore RFC 1122 and RFC 1123, the definition of the TCP/IP model, do not discuss hardware issues and physical data transmission and set no standards for those aspects, other than broadly including them as link-layer components. Some textbook authors[1] [2] have supported the interpretation that physical data transmission aspects are part of the link layer. That position will be held in the rest of this article. Others[3] [4] assumed that physical data transmission standards are not considered as communication protocols, and are not part of the TCP/IP model. These authors assume a hardware layer or physical layer below the link layer, and several of them adopt the OSI term data link layer instead of link layer in a modified description of layering. In the predecessor to the TCP/IP model, the *Arpanet Reference Model* (RFC 908, 1982), aspects of the link layer are referred to by several poorly defined terms, such as *network-access layer*, *network-access protocol*, as well as *network layer*, while the next higher layer is called *internetwork layer*. In some modern text books, *network-interface layer*, *host-to-network layer* and *network-access layer* occur as synonyms either to the link layer or the data link layer, often including the physical layer.

Link-layer protocols

The core protocols specified by the Internet Engineering Task Force to be placed into this layer are the Address Resolution Protocol (ARP), its cousin, the Reverse Address Resolution Protocol (RARP), and the Neighbor Discovery Protocol (NDP), which is a facility delivering similar functionality as ARP for IPv6. Since the advent of IPv6, Open Shortest Path First (OSPF) can be considered to operate on the link level as well, although the IPv4 version of the protocol was considered at the Internet layer.

IS-IS (RFC 1142) is another link-state routing protocol that fits into this layer when considering TCP/IP model, however it was developed within the OSI reference stack (where it is a layer-3 protocol). It is not an Internet standard.

The link layer also contains all hardware specific interface methods, such as Ethernet and other IEEE 802 encapsulation schemes (see References).

Relation to OSI model

The link layer of the TCP/IP model is often compared directly with the combination of the data link layer and the physical layer in the Open Systems Interconnection (OSI) protocol stack. Although they are congruent to some degree in technical coverage of protocols, they are not identical. The link layer in TCP/IP is still wider in scope and in principle a different concept and terminology of classification. This may be observed when certain protocols, such as the Address Resolution Protocol (ARP), which is confined to the link layer in the TCP/IP model, is often said to fit between OSI's data link layer and the network layer. In general, direct or strict comparisons should be avoided, because the layering in TCP/IP is not a principal design criterion and in general considered to be "harmful" (RFC 3439).

Another term sometimes encountered, "network-access layer", tries to suggest the closeness of this layer to the physical network. However, this use is misleading and non-standard, since the link layer implies functions that are wider in scope than just *network access*. Important link-layer protocols are used to probe the topology of the local network, discover routers and neighboring hosts, i.e. functions that go well beyond *network access*.

Examples of functions supported

The link layer includes the following functionality:

- Logical link-local networking methods
  - Encapsulation of IP packets into frames
  - Frame synchronization
  - Error detection with removal of erroneous packets (Used in LANs and point-to-point fiber optical cables)
  - Logical link control (LLC) sublayer: (Used in modem protocols and wireless protocols)
    - Node-to-node error detection and automatic repeat request of erroneous packets
    - Node-to-node flow control
    - Forward error correction (however much more common at the physical layer)
  - Media access control (MAC) sublayer:
    - Multiple access protocols, for example with collision detection or avoidance
    - Physical addressing (MAC addressing)
    - LAN switching (packet switching) including MAC filtering and spanning tree protocol
    - Store-and-forward switching or cut-through switching
    - Data packet queueing or scheduling algorithms
  - Help protocols not encapsulated in IP packets:
    - Some routing protocols
    - IP address to/from physical address resolution protocols
- Physical networking aspects: (It is disputable if and to which extent these are included)
  - Bit-by-bit or symbol-by-symbol delivery
  - Net bit rate
  - Digital modulation
  - Line coding
  - Carrier sense and collision detection utilized by some level 2 multiple-access protocols
  - Circuit switching, establishing circuit switched connections
  - Multiplexing
  - Forward error correction codes
  - Serial or parallel communication
  - Bit synchronization
  - Start-stop signalling and flow control in asynchronous serial communication

- Signal strength (voltage and power levels), impedance
- Forward error correction, bit-interleaving and other channel coding
- Equalization filtering, training sequences, pulse shaping and other signal processing of physical signals
- Mechanical specification of electrical connectors and cables, for example maximum cable length
- Electrical specification of transmission line signal level and impedance
- Radio interface, including electromagnetic spectrum frequency allocation and specification of signal strength, analog bandwidth, etc.
- Specifications for IR over optical fiber or a wireless IR communication link

RFC references

- RFC 1122, "Requirements for Internet Hosts -- Communication layers," IETF, R. Braden (Editor), October 1989
- RFC 1123, "Requirements for Internet Hosts -- Application and Support," IETF, R. Braden (Editor), October 1989
- RFC 893, "Trailer Encapsulations," S. Leffler and M. Karels, April 1984
- RFC 826, "An Ethernet Address Resolution Protocol," D. Plummer, November 1982
- RFC 894, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," C. Hornig, April 1984
- RFC 1042, "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks," J. Postel and J. Reynolds, February 1988
- RFC 2740, "OSPF for IPv6", R. Coltun, et al., December 1999

References

[1] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach, 2007, ISBN 0321497708 (http://www.pearsonhighered.com/educator/academic/product/0,,0321497708,00+en-USS_01DBC.html)

[2] Mark Dye, Mark A. Dye, Wendell, Network Fundamentals: CCNA Exploration Companion Guide, 2007, ISBN 1587132087

[3] Douglas E. Comer, Internetworking with TCP/IP: Principles, Protocols and Architecture, Pearson Prentice Hall 2005, ISBN 0131876716 (http://books.google.com/books?id=jonyuTASbWAC&pg=PA155&hl=sv&source=gbs_toc_r&cad=0_0&sig=ACfU3U18gHAia1pU_Pxn-rhkCnH1v70M6Q#PPA161,M1)

[4] Charles M. Kozierok, "The TCP/IP Guide", No Starch Press 2005 (http://books.google.com/books?id=Pm4RgYV2w4YC&pg=PA131&dq="TCP/IP+model+layers"&lr=&hl=sv&sig=ACfU3U3ofMwYAbZfGz1BmAXc2oNNFC2b8A#PPA129,M1)

External links

- IEEE 802 Standards (http://standards.ieee.org/getieee802/portfolio.html)

Logical link control

The **logical link control** (**LLC**) data communication protocol layer is the upper sub-layer of the data link layer (which is itself layer 2, just above the physical layer) in the seven-layer OSI reference model. It provides multiplexing mechanisms that make it possible for several network protocols (IP, IPX, Decnet and Appletalk) to coexist within a multipoint network and to be transported over the same network media, and can also provide flow control and automatic repeat request (ARQ) error management mechanisms.

The LLC sub-layer acts as an interface between the media access control (MAC) sublayer and the network layer.

Operation

The LLC sublayer is primarily concerned with:

- Multiplexing protocols transmitted over the MAC layer (when transmitting) and decoding them (when receiving).
- Providing node-to-node flow and error control

In today's networks, flow control and error management is typically taken care of by a transport layer protocol such as the TCP protocol, or by some application layer protocol, in an end-to-end fashion, i.e. retransmission is done from source to end destination. This implies that the need for LLC sublayer flow control and error management has reduced. LLC is consequently only a multiplexing feature in today's link layer protocols. An LLC header tells the data link layer what to do with a packet once a frame is received. It works like this: A host will receive a frame and look in the LLC header to find out to what protocol stack the packet is destined - for example, the IP protocol at the network layer or IPX. However, today most non-IP network protocols are abandoned.

Application examples

X.25 and LAPB

An LLC sublayer was a key component in early packet switching networks such as X.25 networks with the LAPB data link layer protocol, where flow control and error management were carried out in a node-to-node fashion, meaning that if an error was detected in a frame, the frame was retransmitted from one switch to next instead. This extensive handshaking between the nodes made the networks slow.

Local area network (LAN) and metropolitan area network (MAN) protocols

The IEEE 802.2 standard specifies LLC sublayer for all IEEE 802 local area networks, such as IEEE 802.3/Ethernet (if the EtherType field isn't used), IEEE 802.5, and IEEE 802.11, and in some non-IEEE 802 networks such as FDDI.

Ethernet

Since bit errors are very rare in wired networks, Ethernet does not provide flow control or automatic repeat request (ARQ), meaning that incorrect packets are detected but only cancelled, not retransmitted (except in case of collisions detected by the CSMA/CD MAC layer protocol). Instead, retransmissions rely on higher layer protocols.

As the Ethertype in an Ethernet II framing formatted frame is used to multiplex different protocols on top of the Ethernet MAC header it can be seen as LLC identifier. However, If Ethernet is used without EtherType field, Ethernet is considered as lacking LLC sublayer.

Wireless LAN

In wireless communications, bit errors are very common. In wireless networks such as IEEE802.11, flow control and error management is part of the CSMA/CA MAC protocol, and not part of the LLC layer. The LLC sublayer follows the IEEE 802.2 standard.

HDLC

Some non-IEEE 802 protocols can be thought of as being split into MAC and LLC layers. For example, while HDLC specifies both MAC functions (framing of packets) and LLC functions (protocol multiplexing, flow control, detection, and error control through a retransmission of dropped packets when indicated), some protocols such as Cisco HDLC can use HDLC-like packet framing and their own LLC protocol.

PPP and modems

Over telephone network modems, PPP link layer protocols can be considered as a LLC protocol, providing multiplexing, but it does not provide flow control and error management. In a telephone network, bit errors might be common, meaning that error management is crucial, but that is today provided by modern modem protocols. Today's modem protocols have inherited LLC features from the older LAPM link layer protocol, made for modem communication in old X.25 networks.

Cellular systems

The GPRS LLC layer also does ciphering and deciphering of SN-PDU (SNDCP) packets.

Power lines

Another example of a data link layer which is split between LLC (for flow and error control) and MAC (for multiple access) is the ITU-T G.hn standard, which provides high-speed local area networking over existing home wiring (power lines, phone lines and coaxial cables).

Media access control

The **media access control** (**MAC**) data communication protocol sub-layer, also known as the *medium access control*, is a sublayer of the data link layer specified in the seven-layer OSI model (layer 2). It provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. Ethernet. The hardware that implements the MAC is referred to as a *medium access controller*.

The MAC sub-layer acts as an interface between the logical link control (LLC) sublayer and the network's physical layer. The MAC layer emulates a full-duplex logical communication channel in a multi-point network. This channel may provide unicast, multicast or broadcast communication service.

Functions performed in the MAC layer

According to 802.3-2002 section 4.1.4, the functions required of a MAC are:[1]

- receive/transmit normal frames
- half-duplex retransmission and backoff functions
- append/check FCS (frame check sequence)
- interframe gap enforcement
- discard malformed frames
- append(tx)/remove(rx) preamble, SFD, and padding
- half-duplex compatibility: append(tx)/remove(rx) MAC address

In 100Mbps and faster MACs, the MAC address is not actually handled in the MAC layer. Doing so would make it impossible to implement IP because the ARP layer of IP-Ethernet needs access to the MAC address.

Addressing mechanism

In 100Mbps and faster Ethernet MACs, there is no required addressing mechanism. However, the MAC address inherited from the original MAC layer specification is used in many higher level protocols such as Internet Protocol (IP) over Ethernet.

The local network address used in IP-Ethernet is called MAC address because it historically was part of the MAC layer in early Ethernet implementations. The MAC layer's addressing mechanism is called physical address or MAC address. A MAC address is a unique serial number. Once a MAC address has been assigned to a particular network interface (typically at time of manufacture), that device should be uniquely identifiable amongst all other network devices in the world. This guarantees that each device in a network will have a different MAC address (analogous to a street address). This makes it possible for data packets to be delivered to a destination within a subnetwork, i.e. hosts interconnected by some combination of repeaters, hubs, bridges and switches, but not by IP routers. Thus, when an IP packet reaches its destination (sub)network, the destination IP address (a layer 3 or network layer concept) is resolved with the Address Resolution Protocol for IPv4, or by Neighbor Discovery Protocol (IPv6) into the MAC address (a layer 2 concept) of the destination host.

An example of a physical network is an Ethernet network, perhaps extended by wireless local area network (WLAN) access points and WLAN network adapters, since these share the same 48-bit MAC address hierarchy as Ethernet.

A MAC layer is not required in full-duplex point-to-point communication, but address fields are included in some point-to-point protocols for compatibility reasons.

Channel access control mechanism

The channel access control mechanisms provided by the MAC layer are also known as a multiple access protocol. This makes it possible for several stations connected to the same physical medium to share it. Examples of shared physical media are bus networks, ring networks, hub networks, wireless networks and half-duplex point-to-point links. The multiple access protocol may detect or avoid data packet collisions if a packet mode contention based channel access method is used, or reserve resources to establish a logical channel if a circuit switched or channelization based channel access method is used. The channel access control mechanism relies on a physical layer multiplex scheme.

The most widespread multiple access protocol is the contention based CSMA/CD protocol used in Ethernet networks. This mechanism is only utilized within a network collision domain, for example an Ethernet bus network or a hub network. An Ethernet network may be divided into several collision domains, interconnected by bridges and switches.

A multiple access protocol is not required in a switched full-duplex network, such as today's switched Ethernet networks, but is often available in the equipment for compatibility reasons.

Common multiple access protocols

Examples of common packet mode multiple access protocols for wired multi-drop networks are:

- CSMA/CD (used in Ethernet and IEEE 802.3)
- Token bus (IEEE 802.4)
- Token ring (IEEE 802.5)
- Token passing (used in FDDI)

Examples of common multiple access protocols that may be used in packet radio wireless networks are:

- CSMA/CA (used in IEEE 802.11/WiFi WLANs)
- Slotted ALOHA
- Dynamic TDMA
- Reservation ALOHA (R-ALOHA)
- Mobile Slotted Aloha (MS-ALOHA)
- CDMA
- OFDMA

References

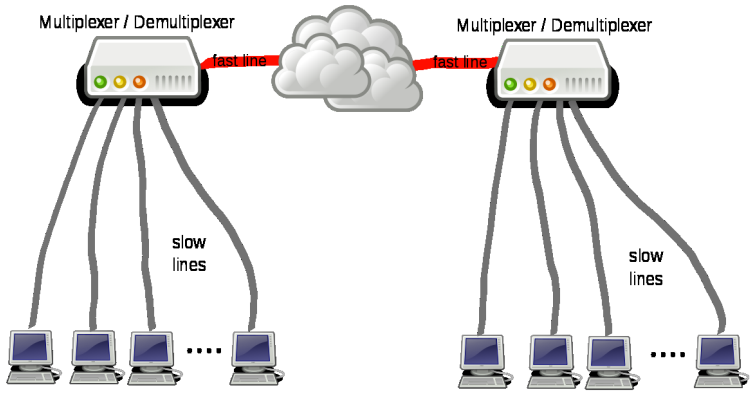
[1] IEEE 802.3

This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.

Multiplexing

For other uses of multiplex, see multiplex (disambiguation).

For multiplexing in electronics and signal processing, see Multiplexer.



General multiplex scheme: the ν input lines-channels are multiplexed into a single fast line. The demultiplexer receives the multiplexed data stream and extracts the original channels to be transferred.

In telecommunications and computer networks, **multiplexing** (also known as **muxing**) is a method by which multiple analog message signals or digital data streams are combined into one signal over a shared medium. The aim is to share an expensive resource. For example, in telecommunications, several telephone calls may be carried using one wire. Multiplexing originated in telegraphy, and is now widely applied in communications.

The multiplexed signal is transmitted over a communication channel, which may be a physical transmission medium. The multiplexing divides the capacity of the low-level communication channel into several higher-level logical channels, one for each message signal or data stream to be transferred. A reverse process, known as demultiplexing, can extract the original channels on the receiver side.

A device that performs the multiplexing is called a multiplexer (MUX), and a device that performs the reverse process is called a demultiplexer (DEMUX).

Inverse multiplexing (IMUX) has the opposite aim as multiplexing, namely to break one data stream into several streams, transfer them simultaneously over several communication channels, and recreate the original data stream.

Types of multiplexing

Multiplexing technologies may be divided into several types, all of which have significant variations:[1] space-division multiplexing (SDM), frequency-division multiplexing (FDM), time-division multiplexing (TDM), and code division multiplexing (CDM). Variable bit rate digital bit streams may be transferred efficiently over a fixed bandwidth channel by means of statistical multiplexing, for example packet mode communication. Packet mode communication is an asynchronous mode time-domain multiplexing which resembles time-division multiplexing.

Digital bit streams can be transferred over an analog channel by means of code-division multiplexing (CDM) techniques such as frequency-hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS).

In wireless communications, multiplexing can also be accomplished through alternating polarization (horizontal/vertical or clockwise/counterclockwise) on each adjacent channel and satellite, or through phased multi-antenna array combined with a Multiple-input multiple-output communications (MIMO) scheme.

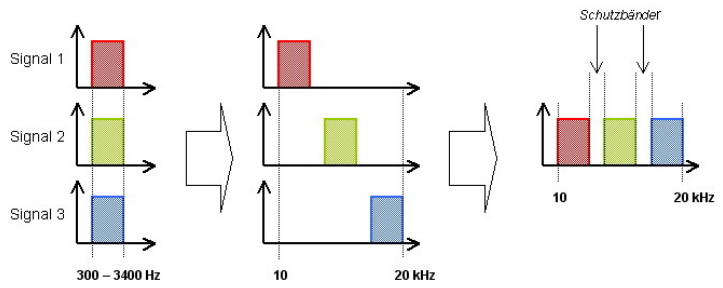
Space-division multiplexing

In wired communication, space-division multiplexing simply implies different point-to-point wires for different channels. Examples include an analogue stereo audio cable, with one pair of wires for the left channel and another for the right channel, and a multipair telephone cable. Another example is a switched star network such as the analog telephone access network (although inside the telephone exchange or between the exchanges, other multiplexing techniques are typically employed) or a switched Ethernet network. A third example is a mesh network. Wired space-division multiplexing is typically not considered as multiplexing.

In wireless communication, space-division multiplexing is achieved by multiple antenna elements forming a phased array antenna. Examples are multiple-input and multiple-output (MIMO), single-input and multiple-output (SIMO) and multiple-input and single-output (MISO) multiplexing. For example, a IEEE 802.11n wireless router with *N* antennas makes it possible to communicate with *N* multiplexed channels, each with a peak bit rate of 54 Mbit/s, thus increasing the total peak bit rate with a factor *N*. Different antennas would give different multi-path propagation (echo) signatures, making it possible for digital signal processing techniques to separate different signals from each other. These techniques may also be utilized for space diversity (improved robustness to fading) or beamforming (improved selectivity) rather than multiplexing.

Frequency-division multiplexing

Frequency-division multiplexing (FDM) is inherently an analog technology. FDM achieves the combining of several digital signals into one medium by sending signals in several distinct frequency ranges over that medium.



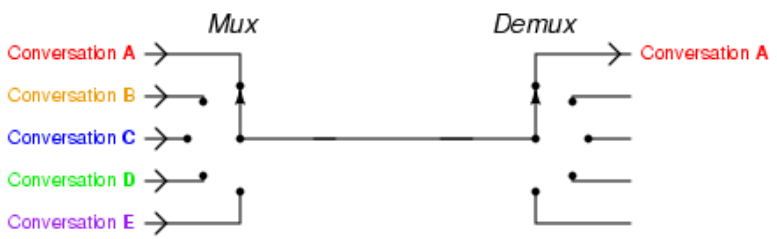
Frequency-division multiplexing (FDM): The spectrums of each input signal are swifted in several distinct frequency ranges.

One of FDM's most common applications is cable television. Only one cable reaches a customer's home but the service provider can send multiple television channels or signals simultaneously over that cable to all subscribers. Receivers must tune to the appropriate frequency (channel) to access the desired signal.[1]

A variant technology, called wavelength-division multiplexing (WDM) is used in optical communications.

Time-division multiplexing

Time-division multiplexing (TDM) is a digital (or in rare cases, analog) technology. TDM involves sequencing groups of a few bits or bytes from each individual input stream, one after the other, and in such a way that they can be associated with the appropriate receiver. If done sufficiently and quickly, the receiving devices will not detect that some of the circuit time was used to serve another logical communication path.



Time-division multiplexing (TDM).

Consider an application requiring four terminals at an airport to reach a central computer. Each terminal communicated at 2400 bit/s, so rather than acquire four individual circuits to carry such a low-speed transmission, the airline has installed a pair of multiplexers. A pair of 9600 bit/s modems and one dedicated analog communications circuit from the airport ticket desk back to the airline data center are also installed.[1]

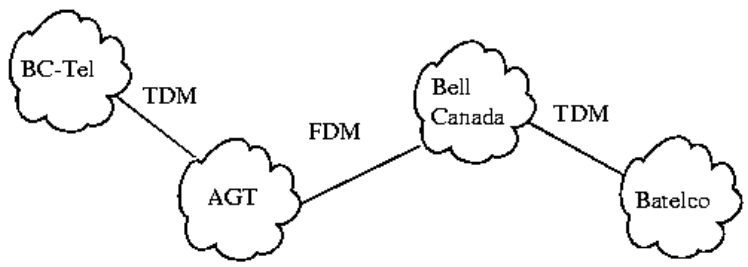
Code-division multiplexing

Code division multiplexing (CDM) or spread spectrum is a class of techniques where several channels simultaneously share the same frequency spectrum, and this spectral bandwdith is much higher than the bit rate or symbol rate. One form is frequency hopping, another is direct sequence spread spectrum. In the latter case, each channel transmits its bits as a coded channel-specific sequence of pulses called chips. Number of chips per bit, or chips per symbol, is the spreading factor. This coded transmission typically is accomplished by transmitting a unique time-dependent series of short pulses, which are placed within chip times within the larger bit time. All channels, each with a different code, can be transmitted on the same fiber or radio channel or other medium, and asynchronously demultiplexed. Advantages over conventional techniques are that variable bandwidth is possible (just as in statistical multiplexing), that the wide bandwidth allows poor signal-to-noise ratio according to Shannon-Heartly theorem, and that multi-path propagation in wireless communication can be combated by rake receivers.

Code Division Multiplex techniques are used as an channel access scheme, namely Code Division Multiple Access (CDMA), e.g. for mobile phone service and in wireless networks, with the advantage of spreading intercell interference among many users. Confusingly, the generic term *Code Division Multiple access* sometimes refers to a specific CDMA based cellular system defined by Qualcomm.

Another important application of CDMA is the Global Positioning System (GPS).

Relation to multiple access



Telecommunication multiplexing

A multiplexing technique may be further extended into a multiple access method or channel access method, for example TDM into Time-division multiple access (TDMA) and statistical multiplexing into carrier sense multiple access (CSMA). A multiple access method makes it possible for several transmitters connected to the same physical medium to share its capacity.

Multiplexing is provided by the Physical Layer of the OSI model, while multiple access also involves a media access control protocol, which is part of the Data Link Layer.

The Transport layer in the OSI model as well as TCP/IP model provides statistical multiplexing of several application layer data flows to/from the same computer.

Application areas

Telegraphy

The earliest communication technology using electrical wires, and therefore sharing an interest in the economies afforded by multiplexing, was the electric telegraph. Early experiments allowed two separate messages to travel in opposite directions simultaneously, first using an electric battery at both ends, then at only one end.

- Émile Baudot developed a time-multiplexing system of multiple Hughes machines in the 1870s.
- In 1874, the quadruplex telegraph developed by Thomas Edison transmitted two messages in each direction simultaneously, for a total of four messages transiting the same wire at the same time.
- Several workers were investigating acoustic telegraphy, a frequency-division multiplexing technique, which led to the invention of the telephone.

Telephony

In telephony, a customer's telephone line now typically ends at the remote concentrator box down the street, where it is multiplexed along with other telephone lines for that neighborhood or other similar area. The multiplexed signal is then carried to the central switching office on significantly fewer wires and for much further distances than a customer's line can practically go. This is likewise also true for digital subscriber lines (DSL).

Fiber in the loop (FITL) is a common method of multiplexing, which uses optical fiber as the backbone. It not only connects POTS phone lines with the rest of the PSTN, but also replaces DSL by connecting directly to Ethernet wired into the home. Asynchronous Transfer Mode is often the communications protocol used.

Because all the phone (and data) lines have been clumped together, none of them can be accessed except through a demultiplexer. Where such demultiplexers are uncommon, this provides for more-secure communications, though the connections are not typically encrypted.

Cable TV has long carried multiplexed television channels, and late in the 20th century began offering the same services as telephone companies. IPTV also depends on multiplexing.

Video processing

In video editing and processing systems, multiplexing refers to the process of interleaving audio and video into one coherent MPEG transport stream (time-division multiplexing).

In digital video, such a transport stream is normally a feature of a container format which may include metadata and other information, such as subtitles. The audio and video streams may have variable bit rate. Software that produces such a transport stream and/or container is commonly called a statistical multiplexor or **muxer**. A **demuxer** is software that extracts or otherwise makes available for separate processing the components of such a stream or container.

Digital broadcasting

In digital television and digital radio systems, several variable bit-rate data streams are multiplexed together to a fixed bitrate transport stream by means of statistical multiplexing. This makes it possible to transfer several video and audio channels simultaneously over the same frequency channel, together with various services.

In the digital television systems, this may involve several standard definition television (SDTV) programmes (particularly on DVB-T, DVB-S2, ISDB and ATSC-C), or one HDTV, possibly with a single SDTV companion channel over one 6 to 8 MHz-wide TV channel. The device that accomplishes this is called a statistical multiplexer. In several of these systems, the multiplexing results in an MPEG transport stream. The newer DVB standards DVB-S2 and DVB-T2 has the capacity to carry several HDTV channels in one multiplex. Even the original DVB standards can carry more HDTV channels in a multiplex if the most advanced MPEG-4 compressions hardware is

used.

On communications satellites which carry broadcast television networks and radio networks, this is known as **multiple channel per carrier** or **MCPC**. Where multiplexing is not practical (such as where there are different sources using a single transponder), single channel per carrier mode is used.

Signal multiplexing of satellite TV and radio channels is typically carried out in a central signal playout and uplink centre, such as ASTRA Platform Services in Germany, which provides playout, digital archiving, encryption, and satellite uplinks, as well as multiplexing, for hundreds of digital TV and radio channels.

In digital radio, both the Eureka 147 system of digital audio broadcasting and the in-band on-channel HD Radio, FMeXtra, and Digital Radio Mondiale systems can multiplex channels. This is essentially required with DAB-type transmissions (where a multiplex is called an **ensemble**), but is entirely optional with IBOC systems.

Analog broadcasting

In FM broadcasting and other analog radio media, multiplexing is a term commonly given to the process of adding subcarriers to the audio signal before it enters the transmitter, where modulation occurs. Multiplexing in this sense is sometimes known as **MPX**, which in turn is also an old term for stereophonic FM, seen on stereo systems since the 1960s.

Other meanings

In spectroscopy the term is used in a related sense to indicate that the experiment is performed with a mixture of frequencies at once and their respective response unravelled afterwards using the Fourier transform principle.

In computer programming, it may refer to using a single in-memory resource (such as a file handle) to handle multiple external resources (such as on-disk files).[2]

Some electrical multiplexing techniques do not require a physical "multiplexer" device, they refer to a "keyboard matrix" or "Charlieplexing" design style:

- Multiplexing may refer to the design of a multiplexed display (non-multiplexed displays are immune to the Dorito effect).
- Multiplexing may refer to the design of a "switch matrix" (non-multiplexed buttons are immune to "phantom keys" and also immune to "phantom key blocking").

References

[1] *Voice and Data Communications*, ISBN 978-0-0-07-225732-9
[2] "Multiplexing filehandles with select() in perl" (http://www.perlfect.com/articles/select.shtml). .

- "Federal Standard 1037C: Glossary of Telecommunications Terms" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm). Institute for Telecommunication Services. Retrieved 2009-10-19.

External links

- Wireless Multiplexing (http://www.entryboot.com/wireless--multiplexing-techniques.php)

Network switch

A **network switch** or **switching hub** is a computer networking device that connects network segments.

The term commonly refers to a multi-port network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (Layer 3) and above are often referred to as Layer 3 switches or multilayer switches.

The first Ethernet switch was introduced by Kalpana in 1990.[1]

Function

The network switch plays an integral part in most modern Ethernet local area networks (LANs). Mid-to-large sized LANs contain a number of linked managed switches. Small office/home office (SOHO) applications typically use a single switch, or an all-purpose converged device such as a residential gateway to access small office/home broadband services such as DSL or cable internet. In most of these cases, the end-user device contains a router and components that interface to the particular physical broadband technology. User devices may also include a telephone interface for VoIP.

An Ethernet switch operates at the data link layer of the OSI model to create a separate collision domain for each switch port. With 4 computers (e.g., A, B, C, and D) on 4 switch ports, A and B can transfer data back and forth, while C and D also do so simultaneously, and the two conversations will not interfere with one another. In the case of a hub, they would all share the bandwidth and run in half duplex, resulting in collisions, which would then necessitate retransmissions. Using a switch is called microsegmentation. This allows computers to have dedicated bandwidth on a point-to-point connections to the network and to therefore run in full duplex without collisions.

Role of switches in networks

Switches may operate at one or more layers of the OSI model, including data link and network. A device that operates simultaneously at more than one of these layers is known as a multilayer switch.

In switches intended for commercial use, built-in or modular interfaces make it possible to connect different types of networks, including Ethernet, Fibre Channel, ATM, ITU-T G.hn and 802.11. This connectivity can be at any of the layers mentioned. While Layer 2 functionality is adequate for bandwidth-shifting within one technology, interconnecting technologies such as Ethernet and token ring is easier at Layer 3.

Devices that interconnect at Layer 3 are traditionally called routers, so "Layer-3 switches" can also be regarded as (relatively primitive) routers.

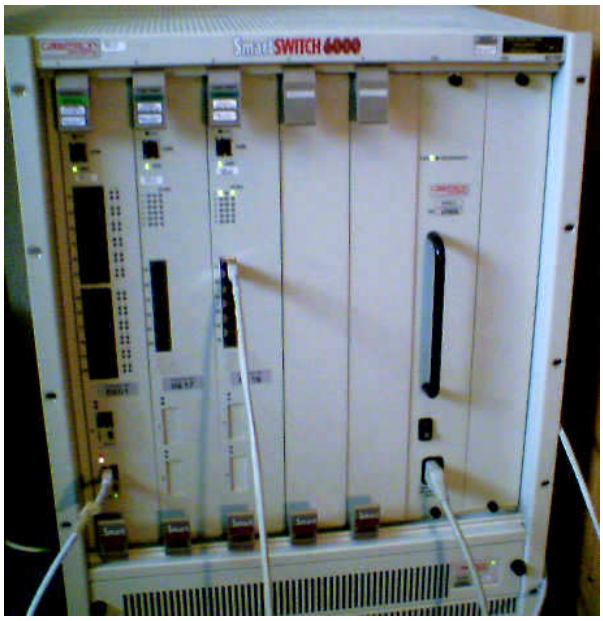
In some service provider and other environments where there is a need for a great deal of analysis of network performance and security, switches may be connected between WAN routers as places for analytic modules. Some vendors provide firewall,[2] [3] network intrusion detection,[4] and performance analysis modules that can plug into switch ports. Some of these functions may be on combined modules.[5]

In other cases, the switch is used to create a mirror image of data that can go to an external device. Since most switch port mirroring provides only one mirrored stream, network hubs can be useful for fanning out data to several read-only analyzers, such as intrusion detection systems and packet sniffers.

Layer-specific functionality

While switches may learn about topologies at many layers, and forward at one or more layers, they do tend to have common features. Other than for high-performance applications, modern commercial switches use primarily Ethernet interfaces, which can have different input and output bandwidths of 10, 100, 1000 or 10,000 megabits per second.

At any layer, a modern switch may implement power over Ethernet (PoE), which avoids the need for attached devices, such as a VoIP phone or wireless access point, to have a separate power supply. Since switches can have redundant power circuits connected to uninterruptible power supplies, the connected device can continue operating even when regular office power fails.



A modular network switch with three network modules (a total of 24 Ethernet and 14 Fast Ethernet ports) and one power supply.

Layer 1 hubs versus higher-layer switches

A network hub, or repeater, is a simple network device. Hubs do not manage any of the traffic that comes through them. Any packet entering a port is broadcast out or "repeated" on every other port, except for the port of entry. Since every packet is repeated on every other port, packet collisions affect the entire network, limiting its capacity.

There are specialized applications where a hub can be useful, such as copying traffic to multiple network sensors. High end switches have a feature which does the same thing called port mirroring.

By the early 2000s, there was little price difference between a hub and a low-end switch.[6]

Layer 2

A network bridge, operating at the data link layer, may interconnect a small number of devices in a home or the office. This is a trivial case of bridging, in which the bridge learns the MAC address of each connected device.

Single bridges also can provide extremely high performance in specialized applications such as storage area networks.

Classic bridges may also interconnect using a spanning tree protocol that disables links so that the resulting local area network is a tree without loops. In contrast to routers, spanning tree bridges must have topologies with only one active path between two points. The older IEEE 802.1D spanning tree protocol could be quite slow, with forwarding stopping for 30 seconds while the spanning tree would reconverge. A Rapid Spanning Tree Protocol was introduced as IEEE 802.1w, but the newest edition of IEEE 802.1D adopts the 802.1w extensions as the base standard.

The IETF is specifying the TRILL protocol, which is the application of link-state routing technology to the layer-2 bridging problem. Devices which implement TRILL, called RBridges, combine the best features of both routers and bridges.

While "layer 2 switch" remains more of a marketing term than a technical term, the products that were introduced as "switches" tended to use microsegmentation and Full duplex to prevent collisions among devices connected to Ethernet. By using an internal forwarding plane much faster than any interface, they give the impression of simultaneous paths among multiple devices.

Once a bridge learns the topology through a spanning tree protocol, it forwards data link layer frames using a layer 2 forwarding method. There are four forwarding methods a bridge can use, of which the second through fourth method were performance-increasing methods when used on "switch" products with the same input and output port bandwidths:

1. Store and forward: The switch buffers and verifies each frame before forwarding it.
2. Cut through: The switch reads only up to the frame's hardware address before starting to forward it. Cut-through switches have to fall back to store and forward if the outgoing port is busy at the time the packet arrives. There is no error checking with this method.
3. Fragment free: A method that attempts to retain the benefits of both store and forward and cut through. Fragment free checks the first 64 bytes of the frame, where addressing information is stored. According to Ethernet specifications, collisions should be detected during the first 64 bytes of the frame, so frames that are in error because of a collision will not be forwarded. This way the frame will always reach its intended destination. Error checking of the actual data in the packet is left for the end device.
4. Adaptive switching: A method of automatically selecting between the other three modes.

While there are specialized applications, such as storage area networks, where the input and output interfaces are the same bandwidth, this is rarely the case in general LAN applications. In LANs, a switch used for end user access typically concentrates lower bandwidth (e.g., 10/100 Mbit/s) into a higher bandwidth (at least 1 Gbit/s). Alternatively, a switch that provides access to server ports usually connects to them at a much higher bandwidth than is used by end user devices.

Layer 3

Within the confines of the Ethernet physical layer, a layer 3 switch can perform some or all of the functions normally performed by a router. The most common layer-3 capability is awareness of IP multicast through IGMP snooping. With this awareness, a layer-3 switch can increase efficiency by delivering the traffic of a multicast group only to ports where the attached device has signaled that it wants to listen to that group.

Layer 4

While the exact meaning of the term Layer-4 switch is vendor-dependent, it almost always starts with a capability for network address translation, but then adds some type of load distribution based on TCP sessions.[7]

The device may include a stateful firewall, a VPN concentrator, or be an IPSec security gateway.

Layer 7

Layer 7 switches may distribute loads based on URL or by some installation-specific technique to recognize application-level transactions. A Layer-7 switch may include a web cache and participate in a content delivery network.[8]

Types of switches

Rack-mounted 24-port 3Com switch

Form factor

- Desktop, not mounted in an enclosure, typically intended to be used in a home or office environment outside of a wiring closet
- Rack mounted - A switch that mounts in an equipment rack
- Chassis - with swappable module cards
- DIN rail mounted - normally seen in industrial environments or panels

Configuration options

- *Unmanaged* switches — These switches have no configuration interface or options. They are plug and play. They are typically the least expensive switches, found in home, SOHO, or small businesses. They can be desktop or rack mounted.
- *Managed* switches — These switches have one or more methods to modify the operation of the switch. Common management methods include: a command-line interface (CLI) accessed via serial console, telnet or Secure Shell, an embedded Simple Network Management Protocol (SNMP) agent allowing management from a remote console or management station, or a web interface for management from a web browser. Examples of configuration changes that one can do from a managed switch include: enable features such as Spanning Tree Protocol, set port bandwidth, create or modify Virtual LANs (VLANs), etc. Two sub-classes of managed switches are marketed today:
  - *Smart* (or intelligent) switches — These are managed switches with a limited set of management features. Likewise "web-managed" switches are switches which fall in a market niche between unmanaged and managed. For a price much lower than a fully managed switch they provide a web interface (and usually no CLI access) and allow configuration of basic settings, such as VLANs, port-bandwidth and duplex.[9]
  - *Enterprise Managed* (or fully managed) switches — These have a full set of management features, including CLI, SNMP agent, and web interface. They may have additional features to manipulate configurations, such as the ability to display, modify, backup and restore configurations. Compared with smart switches, enterprise switches have more features that can be customized or optimized, and are generally more expensive than smart switches. Enterprise switches are typically found in networks with larger number of switches and connections, where centralized management is a significant savings in administrative time and effort. A stackable switch is a version of enterprise-managed switch.

Traffic monitoring on a switched network

Unless port mirroring or other methods such as RMON, SMON or sFlow are implemented in a switch,[10] it is difficult to monitor traffic that is bridged using a switch because only the sending and receiving ports can see the traffic. These monitoring features are rarely present on consumer-grade switches.

Two popular methods that are specifically designed to allow a network analyst to monitor traffic are:

- Port mirroring — the switch sends a copy of network packets to a monitoring network connection.
- SMON — "Switch Monitoring" is described by RFC 2613 and is a protocol for controlling facilities such as port mirroring.

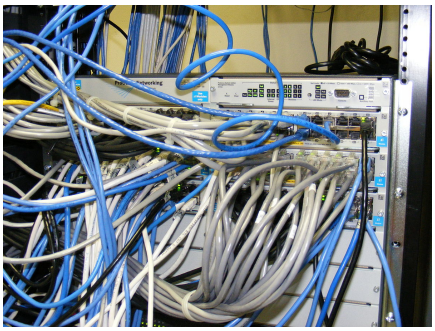
Another method to monitor may be to connect a Layer-1 hub between the monitored device and its switch port. This will induce minor delay, but will provide multiple interfaces that can be used to monitor the individual switch port.

Typical switch management features

- Turn particular port range on or off
- Link bandwidth and duplex settings
- Priority settings for ports
- IP Management by IP Clustering.
- MAC filtering and other types of "port security" features which prevent MAC flooding
- Use of Spanning Tree Protocol
- SNMP monitoring of device and link health
- Port mirroring (also known as: port monitoring, spanning port, SPAN port, roving analysis port or link mode port)
- Link aggregation (also known as *bonding*, *trunking* or *teaming*)
- VLAN settings
- 802.1X network access control
- IGMP snooping

Linksys 48-port switch

Link aggregation allows the use of multiple ports for the same connection achieving higher data transfer rates. Creating VLANs can serve security and performance goals by reducing the size of the broadcast domain.



HP Procurve rack-mounted switches mounted in a standard Telco Rack 19-inch rack with network cables

References

[1] Robert J. Kohlhepp (2000-10-02). "The 10 Most Important Products of the Decade" (http://www.networkcomputing.com/1119/1119f1products_5.html). Network Computing. . Retrieved 2008-02-25.

[2] Cisco Catalyst 6500 Series Firewall Services Module (http://cisco.com/en/US/products/hw/modules/ps2706/ps4452/index.html), Cisco Systems,2007

[3] Switch 8800 Firewall Module (http://www.3com.com/products/en_US/detail.jsp?tab=features&sku=3C17546&pathtype=purchase3Com), 3Com Corporation, 2006

[4] Cisco Catalyst 6500 Series Intrusion Detection System (IDSM-2) Module (http://cisco.com/en/US/products/hw/modules/ps2706/ps5058/index.html), Cisco Systems,2007

[5] Getting Started with Check Point Fire Wall-1 (http://www.checkpoint.com/support/technical/online_ug/firewall-14.0/config.htm), Checkpoint Software Technologies Ltd., n.d.
[6] Matthew Glidden (October 2001). "Switches and Hubs" (http://www.atpm.com/7.10/networking-1.shtml). *About This Particular Macintosh blog*. . Retrieved June 9, 2011.
[7] S. Sathaye (January 1999), *The Ins and Outs of Layer 4+ Switching* (http://www.nanog.org/mtg-9901/ppt/alteon/alteon.ppt), NANOG 15, , "It usually means one of two things: - 1. Layer 4 information is used to prioritize and queue traffic (routers have done this for years) - 2. Layer 4 information is used to direct application sessions to different servers (next generation load balancing)."
[8] How worried is too worried? Plus, a Global Crossing Story. (http://www.irbs.net/internet/nanog/0110/0618.html), NANOG mailing list archives, S. Gibbard,October 2001
[9] Tech specs for a sample HP "web-managed" switch (archived page from web.archive.org) (http://web.archive.org/web/20071213185114/http://www.hp.com/rnd/products/switches/ProCurve_Switch_1800_Series/specs.htm)
[10] Remote Network Monitoring Management Information Base (http://www.ietf.org/rfc/rfc2819.txt), RFC 2819, S. Waldbusser,May 2000

External links

- Five datacenter switches compared -- Network World, 2010 (http://www.networkworld.com/community/node/62095)

Packet switching

Packet switching is a digital networking communications method that groups all transmitted data – regardless of content, type, or structure – into suitably sized blocks, called *packets*. Packet switching features delivery of variable-bit-rate data streams (sequences of packets) over a shared network. When traversing network adapters, switches, routers and other network nodes, packets are buffered and queued, resulting in variable delay and throughput depending on the traffic load in the network.

Packet switching contrasts with another principal networking paradigm, circuit switching, a method which sets up a limited number of dedicated connections of constant bit rate and constant delay between nodes for exclusive use during the communication session. In case of traffic fees (as opposed to flate rate), for example in cellular communication services, circuit switching is characterized by a fee per time unit of connection time, even when no data is transferred, while packet switching is characterized by a fee per unit of information.

Two major packet switching modes exist; (1) connectionless packet switching, also known as datagram switching, and (2) connection-oriented packet switching, also known as virtual circuit switching. In the first case each packet includes complete addressing or routing information. The packets are routed individually, sometimes resulting in different paths and out-of-order delivery. In the second case a connection is defined and preallocated in each involved node during a connection phase before any packet is transferred. The packets include a connection identifier rather than address information, and are delivered in order. See below.

Packet mode communication may be utilized with or without intermediate forwarding nodes (packet switches or routers). In all packet mode communication, network resources are managed by statistical multiplexing or dynamic bandwidth allocation in which a communication channel is effectively divided into an arbitrary number of logical variable-bit-rate channels or data streams. Statistical multiplexing, packet switching and other store-and-forward buffering introduces varying latency and throughput in the transmission. Each logical stream consists of a sequence of packets, which normally are forwarded by the multiplexers and intermediate network nodes asynchronously using first-in, first-out buffering. Alternatively, the packets may be forwarded according to some scheduling discipline for fair queuing, traffic shaping or for differentiated or guaranteed quality of service, such as weighted fair queuing or leaky bucket. In case of a shared physical medium, the packets may be delivered according to some packet-mode multiple access scheme.

History

The concept of switching small blocks of data was first explored by Paul Baran in the early 1960s. Independently, Donald Davies at the National Physical Laboratory (NPL) in the UK had developed the same ideas (Abbate, 2000).

Leonard Kleinrock conducted early research in queueing theory which would be important in packet switching, and published a book in the related field of digital message switching (without the packets) in 1961; he also later played a leading role in building and management of the world's first packet-switched network, the ARPANET.

Baran developed the concept of message block switching during his research at the RAND Corporation for the US Air Force into survivable communications networks, first presented to the Air Force in the summer of 1961 as briefing B-265[1] then published as RAND Paper P-2626 [2] in 1962 and then including and expanding somewhat within a series of eleven papers titled On Distributed Communications [3] in 1964. Baran's P-2626 paper described a general architecture for a large-scale, distributed, survivable communications network. The paper focuses on three key ideas: first, use of a decentralized network with multiple paths between any two points; and second, dividing complete user messages into what he called *message blocks* (later called packets); then third, delivery of these messages by store and forward switching.

Baran's study made its way to Robert Taylor and J.C.R. Licklider at the Information Processing Technology Office, both wide-area network evangelists, and it helped influence Lawrence Roberts to adopt the technology when Taylor put him in charge of development of the ARPANET.

Baran's work was similar to the research performed independently by Donald Davies at the National Physical Laboratory, UK. In 1965, Davies developed the concept of packet-switched networks and proposed development of a UK wide network. He gave a talk on the proposal in 1966, after which a person from the Ministry of Defence (MoD) told him about Baran's work. A member of Davies' team met Lawrence Roberts at the 1967 ACM Symposium on Operating System Principles, bringing the two groups together.

Interestingly, Davies had chosen some of the same parameters for his original network design as Baran, such as a packet size of 1024 bits. In 1966 Davies proposed that a network should be built at the laboratory to serve the needs of NPL and prove the feasibility of packet switching. The NPL Data Communications Network entered service in 1970. Roberts and the ARPANET team took the name "packet switching" itself from Davies's work.

The first computer network and packet switching network deployed for computer resource sharing was the Octopus Network at the Lawrence Livermore National Laboratory that began connecting four Control Data 6600 computers to several shared storage devices (including an IBM 2321 Data Cell[4] in 1968 and an IBM Photostore[5] in 1970) and to several hundred Teletype Model 33 ASR terminals for time sharing use starting in 1968.[6]

In 1973 Vint Cerf and Bob Kahn wrote the specifications for Transmission Control Protocol (TCP), an internetworking protocol for sharing resources using packet-switching among the nodes.

Connectionless and connection-oriented packet switching

The service actually provided to the user by networks using packet switching nodes can be either connectionless (based on datagram messages), or virtual circuit switching (also known as connection oriented). Some connectionless protocols are Ethernet, IP, and UDP; connection oriented packet-switching protocols include X.25, Frame relay, Multiprotocol Label Switching (MPLS), and TCP.

In connection-oriented networks, each packet is labeled with a connection ID rather than an address. Address information is only transferred to each node during a connection set-up phase, when the route to the destination is discovered and an entry is added to the switching table in each network node through which the connection passes. The signalling protocols used allow the application to specify its requirements and the network to specify what capacity etc. is available, and acceptable values for service parameters to be negotiated. Routing a packet is very simple, as it just requires the node to look up the ID in the table. The packet header can be small, as it only needs to contain the ID and any information (such as length, timestamp, or sequence number) which is different for different

packets.

In connectionless networks, each packet is labeled with a destination address, source address, and port numbers; it may also be labeled with the sequence number of the packet. This precludes the need for a dedicated path to help the packet find its way to its destination, but means that much more information is needed in the packet header, which is therefore larger, and this information needs to be looked up in power-hungry content-addressable memory. Each packet is dispatched and may go via different routes; potentially, the system has to do as much work for every packet as the connection-oriented system has to do in connection set-up, but with less information as to the application's requirements. At the destination, the original message/data is reassembled in the correct order, based on the packet sequence number. Thus a virtual connection, also known as a virtual circuit or byte stream is provided to the end-user by a transport layer protocol, although intermediate network nodes only provides a connectionless network layer service.

Packet switching in networks

Packet switching is used to optimize the use of the channel capacity available in digital telecommunication networks such as computer networks, to minimize the transmission latency (the time it takes for data to pass across the network), and to increase robustness of communication.

The most well-known use of packet switching is the Internet and most local area networks. The Internet is implemented by the Internet Protocol Suite using a variety of Link Layer technologies. For example, Ethernet and Frame Relay are common. Newer mobile phone technologies (e.g., GPRS, I-mode) also use packet switching.

X.25 is a notable use of packet switching in that, despite being based on packet switching methods, it provided virtual circuits to the user. These virtual circuits carry variable-length packets. In 1978, X.25 provided the first international and commercial packet switching network, the International Packet Switched Service (IPSS). Asynchronous Transfer Mode (ATM) also is a virtual circuit technology, which uses fixed-length cell relay connection oriented packet switching.

Datagram packet switching is also called connectionless networking because no connections are established. Technologies such as Multiprotocol Label Switching (MPLS) and the resource reservation protocol (RSVP) create virtual circuits on top of datagram networks. Virtual circuits are especially useful in building robust failover mechanisms and allocating bandwidth for delay-sensitive applications.

MPLS and its predecessors, as well as ATM, have been called "fast packet" technologies. MPLS, indeed, has been called "ATM without cells".[7] Modern routers, however, do not require these technologies to be able to forward variable-length packets at multigigabit speeds across the network.

X.25 vs. Frame Relay packet switching

Both X.25 and Frame Relay provide connection-oriented packet switching, also known as virtual circuit switching. A major difference between X.25 and Frame Relay packet switching are that X.25 is a reliable protocol, based on node-to-node automatic repeat request, while Frame Relay is a non-reliable protocol, maximum packet length is 1000 bytes. Any retransmissions must be carried out by higher layer protocols. The X.25 protocol is a network layer protocol, and is part of the X.25 protocol suite, also known as the OSI protocol suite. It was widely used in switching networks during the 1980s and early 1990s, for example as an alternative to circuit mode terminal switching, and for automated teller machines. Frame relay is a further development of X.25. The simplicity of Frame Relay made it considerably faster and more cost effective than X.25 packet switching. Frame relay is a data link layer protocol, and does not provide logical addresses and routing. It is only used for "semi-permanent" connections, while X.25 connections also can be established for each communication session. Frame Relay was used to interconnect LANs or LAN segments, mainly in the 1990s by large companies that had a requirement to handle heavy telecommunications traffic across wide area networks.[8] :250 Despite the benefits of frame relay packet switching, many international

companies are staying with the X.25 standard. In the United States, X.25 packet switching was used heavily in government and financial networks that use mainframe applications. Many companies did not intend to cross over to Frame Relay packet switching because it is more cost effective to use X.25 on slower networks. In certain parts of the world, particularly in Asia-Pacific and South America regions, X.25 was the only technology available.[9]

References

[1] Stewart, Bill (2000-01-07). "Paul Baran Invents Packet Switching" (http://www.livinginternet.com/i/ii_rand.htm). *Living Internet*. . Retrieved 2008-05-08.

[2] http://www.rand.org/pubs/papers/P2626/

[3] http://www.rand.org/pubs/research_memoranda/RM3420/index.html

[4] The IBM 2321 Data Cell Drive (http://www.columbia.edu/acis/history/datacell.html), Columbia University Computing History

[5] The IBM 1360 Photostore (http://www.computer-history.info/Page4.dir/pages/Photostore.dir/index.html), Lawrence Livermore Laboratory Computing History

[6] Mendicino, Samuel (1970-11-30). "Octopus: The Lawrence Radiation Laboratory Network" (http://www.rogerdmoore.ca/PS/OCTOA/OCTO.html). . Retrieved 2009-05-06.

[7] Interview with the author (of an MPLS-based VPN article) (http://www.certificationzone.com/cisco/newsletter/SL/interview_08-12-03.html), G. Pildush

[8] O'Brien, J. A. & Marakas, G. M. (2009). Management Information Systems (9th ed.). New York: McGraw-Hill/Irwin.

[9] Girard, K. (1997, January). X.25 users remaining loyal despite frame-relay hype. Computerworld, 31(4), 16. Retrieved March 6, 2009, from ABI/INFORM Global database. (Document ID: 10946641).

Bibliography

- Leonard Kleinrock, Information Flow in Large Communication Nets (http://www.lk.cs.ucla.edu/bibliography-public_reports.html), (MIT, Cambridge, May 31, 1961) Proposal for a Ph.D. Thesis
- Leonard Kleinrock. *Information Flow in Large Communication Nets* (RLE Quarterly Progress Report, July 1961)
- Leonard Kleinrock. *Communication Nets: Stochastic Message Flow and Delay* (McGraw-Hill, New York, 1964)
- Paul Baran et al., *On Distributed Communications, Volumes I-XI* (http://www.rand.org/about/history/baran-list.html) (RAND Corporation Research Documents, August, 1964)
  - Paul Baran, *On Distributed Communications: I Introduction to Distributed Communications Network* (http://www.rand.org/publications/RM/RM3420/) (RAND Memorandum RM-3420-PR. August 1964)
- Paul Baran, On Distributed Communications Networks (http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1088883), (IEEE Transactions on Communications Systems, March 1964)
- D. W. Davies, K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, *A digital communications network for computers giving rapid response at remote terminals* (ACM Symposium on Operating Systems Principles. October 1967)
- R. A. Scantlebury, P. T. Wilkinson, and K. A. Bartlett, *The design of a message switching Centre for a digital communication network* (IFIP 1968)
- Larry Roberts and Tom Merrill, *Toward a Cooperative Network of Time-Shared Computers* (http://www.packet.cc/files/toward-coop-net.html) (Fall AFIPS Conference. October 1966)
- Lawrence Roberts, *The Evolution of Packet Switching* (http://www.packet.cc/files/ev-packet-sw.html) (Proceedings of the IEEE, November, 1978)

Further reading

- Katie Hafner, *Where Wizards Stay Up Late* (Simon and Schuster, 1996) pp 52–67
- Janet Abbate, *Inventing the Internet* (MIT Press, 2000) ISBN 0-262-51115-0
- Arthur Norberg, Judy E. O'Neill, *Transforming Computer Technology: Information Processing for the Pentagon, 1962-1982* (Johns Hopkins University, 1996)

External links

- Oral history interview with Paul Baran (http://www.cbi.umn.edu/oh/display.phtml?id=110). Charles Babbage Institute University of Minnesota, Minneapolis. Baran describes his working environment at RAND, as well as his initial interest in survivable communications, and the evolution, writing and distribution of his eleven-volume work, "On Distributed Communications." Baran discusses his interaction with the group at ARPA who were responsible for the later development of the ARPANET.
- Packet Switching History and Design (http://www.livinginternet.com/i/iw_packet.htm), site reviewed by Baran, Roberts, and Kleinrock
- Paul Baran and the Origins of the Internet (http://www.rand.org/about/history/baran.html)
- A Brief History of the Internet (http://www.isoc.org/internet/history/brief.shtml)

This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.

Point-to-point protocol

In networking, the **Point-to-Point Protocol** (**PPP**) is a data link protocol commonly used in establishing a direct connection between two networking nodes. It can provide connection authentication, transmission encryption, and compression.

PPP is used over many types of physical networks including serial cable, phone line, trunk line, cellular telephone, specialized radio links, and fiber optic links such as SONET. PPP is also used over Internet access connections (now marketed as "broadband"). Internet service providers (ISPs) have used PPP for customer dial-up access to the Internet, since IP packets cannot be transmitted over a modem line on their own, without some data link protocol. Two encapsulated forms of PPP, Point-to-Point Protocol over Ethernet (PPPoE) and Point-to-Point Protocol over ATM (PPPoA), are used most commonly by Internet Service Providers (ISPs) to establish a Digital Subscriber Line (DSL) Internet service connection with customers.

PPP is commonly used as a data link layer protocol for connection over synchronous and asynchronous circuits, where it has largely superseded the older Serial Line Internet Protocol (SLIP) and telephone company mandated standards (such as Link Access Protocol, Balanced (LAPB) in the X.25 protocol suite). PPP was designed to work with numerous network layer protocols, including Internet Protocol (IP), TRILL, Novell's Internetwork Packet Exchange (IPX), NBF and AppleTalk.

Description

PPP and TCP/IP protocol stack

Application	FTP	SMTP	HTTP	…	DNS	…
Transport	TCP				UDP	
Internet	IP				IPv6	
Network access	PPP					
	PPPoE				PPPoA	
	Ethernet				ATM	

PPP was designed somewhat after the original HDLC specifications. The designers of PPP included many additional features that had been seen only in proprietary data-link protocols up to that time.

RFC 2516 describes Point-to-Point Protocol over Ethernet (PPPoE) as a method for transmitting PPP over Ethernet that is sometimes used with DSL. RFC 2364 describes Point-to-Point Protocol over ATM (PPPoA) as a method for transmitting PPP over ATM Adaptation Layer 5 (AAL5), which is also a common alternative to PPPoE used with DSL.

PPP is specified in RFC 1661.

Automatic self configuration

Link Control Protocol (LCP) initiates and terminates connections gracefully, allowing hosts to negotiate connection options. It is an integral part of PPP, and is defined in the same standard specification. LCP provides automatic configuration of the interfaces at each end (such as setting datagram size, escaped characters, and magic numbers) and for selecting optional authentication. The LCP protocol runs on top of PPP (with PPP protocol number 0xC021) and therefore a basic PPP connection has to be established before LCP is able to configure it.

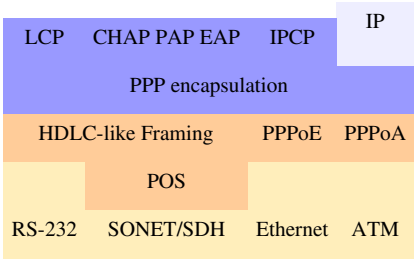
RFC 1994 describes Challenge-handshake authentication protocol (CHAP), which is preferred for establishing dial-up connections with ISPs. Although deprecated, Password authentication protocol (PAP) is still sometimes used.

Another option for authentication over PPP is Extensible Authentication Protocol (EAP) described in RFC 2284.

After the link has been established, additional network (layer 3) configuration may take place. Most commonly, the Internet Protocol Control Protocol (IPCP) is used, although Internetwork Packet Exchange Control Protocol (IPXCP) and AppleTalk Control Protocol (ATCP) were once very popular. Internet Protocol Version 6 Control Protocol (IPv6CP) will see extended use in the future, when IPv6 replaces IPv4's position as the dominant layer-3 protocol.

Multiple network layer protocols

PPP architecture



PPP permits multiple network layer protocols to operate on the same communication link. For every network layer protocol used, a separate **Network Control Protocol** (NCP) is provided in order to encapsulate and negotiate options for the multiple network layer protocols. It negotiates network-layer information, e.g. network address or compression options, after the connection has been established.

For example, Internet Protocol (IP) uses the IP Control Protocol (IPCP), and Internetwork Packet Exchange (IPX) uses the Novell IPX Control Protocol (**IPX/SPX**). NCPs include fields containing standardized codes to indicate the network layer protocol type that the PPP connection encapsulates.

Looped link detection

PPP detects looped links using a feature involving magic numbers. When the node sends PPP LCP messages, these messages may include a magic number. If a line is looped, the node receives an LCP message with its own magic number, instead of getting a message with the peer's magic number.

PPP Configuration Options

The previous section introduced the use of LCP options to meet specific WAN connection requirements. PPP may include the following LCP options:

- **Authentication** - Peer routers exchange authentication messages. Two authentication choices are Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP). Authentication is explained in the next section.
- **Compression** - Increases the effective throughput on PPP connections by reducing the amount of data in the frame that must travel across the link. The protocol decompresses the frame at its destination. See RFC 1962 for more details.
- **Error detection** - Identifies fault conditions. The Quality and Magic Number options help ensure a reliable, loop-free data link. The Magic Number field helps in detecting links that are in a looped-back condition. Until the Magic-Number Configuration Option has been successfully negotiated, the Magic-Number must be transmitted as zero. Magic numbers are generated randomly at each end of the connection.
- **Multilink** - Provides load balancing several interfaces used by PPP through Multilink PPP (see below).

PPP frame

Structure of a PPP frame

Name	Number of bytes	Description
Protocol	1 or 2	setting of protocol in data field
Information	variable (0 or more)	datagram
Padding	variable (0 or more)	optional padding

The Protocol field indicates the type of payload packet (e.g. LCP, NCP, IP, IPX, AppleTalk, etc.).

The Information field contains the PPP payload; it has a variable length with a negotiated maximum called the Maximum Transmission Unit. By default, the maximum is 1500 octets. It might be padded on transmission; if the information for a particular protocol can be padded, that protocol must allow information to be distinguished from padding.

Encapsulation

PPP frames are encapsulated in a lower-layer protocol that provides framing and may provide other functions such as a checksum to detect transmission errors. PPP on serial links is usually encapsulated in a framing similar to HDLC, described by IETF RFC 1662.

Name	Number of bytes	Description
Flag	1	indicates frame's begin or end
Address	1	broadcast address
Control	1	control byte
Protocol	1 or 2	setting of protocol in information field
Information	variable (0 or more)	datagram
Padding	variable (0 or more)	optional padding
FCS	2 (or 4)	error check

The Flag field is present when PPP with HDLC-like framing is used.

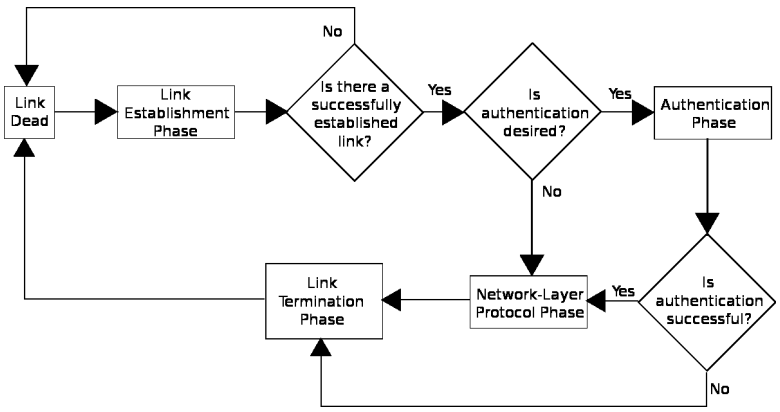
The Address and Control fields always have the value hex FF (for "all stations") and hex 03 (for "unnumbered information"), and can be omitted whenever PPP LCP Address-and-Control-Field-Compression (ACFC) is negotiated.

The Frame Check Sequence (FCS) field is used for determining whether an individual frame has an error. It contains a checksum computed over the frame to provide basic protection against errors in transmission. This is a CRC code similar to the one used for other layer two protocol error protection schemes such as the one used in Ethernet. According to RFC 1662, it can be either 16 bits (2 bytes) or 32 bits (4 bytes) in size (default is 16 bits - Polynomial $x^{16} + x^{12} + x^5 + 1$).

The FCS is calculated over the Address, Control, Protocol, Information and Padding fields after the message have been escaped.

PPP line activation and phases

The phases of the Point to Point Protocol according to RFC 1661 are listed below:



A diagram depicting the phases of PPP according to RFC 1661.

- **Link Dead**. This phase occurs when the link fails, or one side has been told not to connect (e.g. a user has finished his or her dialup connection.)
- **Link Establishment Phase**. This phase is where Link Control Protocol negotiation is attempted. If successful, control goes either to the authentication phase or the Network-Layer Protocol phase, depending on whether authentication is desired.
- **Authentication Phase**. This phase is optional. It allows the sides to authenticate each other before a connection is established. If successful, control goes to the network-layer protocol phase.
- **Network-Layer Protocol Phase**. This phase is where each desired protocols' Network Control Protocols are invoked. For example, IPCP is used in establishing IP service over the line. Data transport for all protocols which are successfully started with their network control protocols also occurs in this phase. Closing down of network protocols also occur in this phase.
- **Link Termination Phase**. This phase closes down this connection. This can happen if there is an authentication failure, if there are so many checksum errors that the two parties decide to tear down the link automatically, if the link suddenly fails, or if the user decides to hang up his connection.

PPP over several links

Multilink PPP

Multilink PPP (also referred to as **MLPPP**, **MP**, **MPPP**, **MLP**, or Multilink) provides a method for spreading traffic across multiple distinct PPP connections. It is defined in RFC 1990. It can be used, for example, to connect a home computer to an Internet Service Provider using two traditional 56k modems, or to connect a company through two leased lines.

On a single PPP line frames cannot arrive out of order, but this is possible when the frames are divided among multiple PPP connections. Therefore Multilink PPP must number the fragments so they can be put in the right order again when they arrive.

Multilink PPP is an example of a link aggregation technology. Cisco IOS Release 11.1 and later supports Multilink PPP.

Multiclass PPP

With PPP, one cannot establish several simultaneous distinct PPP connections over a single link.

That's not possible with Multilink PPP either. Multilink PPP uses contiguous numbers for all the fragments of a packet, and as a consequence it is not possible to suspend the sending of a sequence of fragments of one packet in order to send another packet. This prevents from running Multilink PPP multiple times on the same links.

Multiclass PPP is a kind of Multilink PPP where each "class" of traffic uses a separate sequence number space and reassembly buffer. Multiclass PPP is defined in RFC 2686.

PPP and tunnels

Simplified OSI protocol stack for an example SSH+PPP tunnel

Application	FTP	SMTP	HTTP	…	DNS	…
Transport	TCP				UDP	
Network	IP					
Data Link	**PPP**					
Application	**SSH**					
Transport	TCP					
Network	IP					
Data Link	Ethernet				ATM	
Physical	Cables, NICs, and so on					

Derived protocols

PPTP is a form of PPP between two hosts via GRE using encryption (MPPE) and compression (MPPC).

PPP as a layer 2 protocol between both ends of a tunnel

Many protocols can be used to tunnel data over IP networks. Some of them, like SSL, SSH, or L2TP create virtual network interfaces and give the impression a direct physical connections between the tunnel endpoints. On a Linux host for example, these interfaces would be called **tun0**.

As there are only two endpoints on a tunnel, the tunnel is a point-to-point connection and PPP is a natural choice as a data link layer protocol between the virtual network interfaces. PPP can assign IP addresses to these virtual interfaces, and these IP addresses can be used, for example, to route between the networks on both sides of the tunnel.

IPsec in tunneling mode does not create virtual physical interfaces at the end of the tunnel, since the tunnel is handled directly by the TCP/IP stack. L2TP can be used to provide these interfaces, this technique is called L2TP/IPsec. In this case too, PPP provides IP addresses to the extremities of the tunnel.

RFCs

PPP is defined in RFC 1661 (The Point-to-Point Protocol, July 1994). RFC 1547 (Requirements for an Internet Standard Point-to-Point Protocol, December 1993) provides historical information about the need for PPP and its development. A series of related RFCs have been written to define how a variety of network control protocols-including TCP/IP, DECnet, AppleTalk, IPX, and others-work with PPP.

- RFC 1661, Standard 51, The Point-to-Point Protocol (PPP)
- RFC 1662, Standard 51, PPP in HDLC-like Framing
- RFC 1717, The PPP Multilink Protocol (MP) (obsoleted by RFC 1990)
- RFC 1990, The PPP Multilink Protocol (MP)
- RFC 1994, PPP Challenge Handshake Authentication Protocol (CHAP)
- RFC 2686, The Multi-Class Extension to Multi-Link PPP
- RFC 2153, Informational, PPP Vendor Extensions
- RFC 2284, PPP Extensible Authentication Protocol (EAP)
- RFC 2615, PPP over SONET/SDH
- RFC 2687, Proposed Standard, PPP in a Real-time Oriented HDLC-like Framing
- RFC 5072, IP Version 6 over PPP
- RFC 5172, Negotiation for IPv6 Datagram Compression Using IPv6 Control Protocol

References

External links

- PPP animation (http://www.visualland.net/view.php?cid=1410&protocol=PPP&title=1.Connection- link, IP&ctype=2)

Promiscuous mode

In computer networking, **promiscuous mode** or *promisc mode* is a mode for a network interface controller (NIC) that causes the NIC to pass all traffic it receives to the central processing unit (CPU) rather than just passing frames the NIC is intended to receive. This mode is normally used for packet sniffing and bridged networking for hardware virtualization.

In IEEE 802 networks such as Ethernet, token ring, and Wi-Fi, and in FDDI, each frame includes a destination Media Access Control address (MAC address). In non-promiscuous mode, when a NIC receives a frame, it normally drops it unless the frame is addressed to that NIC's MAC address or is a broadcast or multicast frame. In promiscuous mode, however, the card allows all frames through, thus allowing the computer to read frames intended for other machines or network devices.

Many operating systems require superuser privileges to enable promiscuous mode. A non-routing node in promiscuous mode can generally only monitor traffic to and from other nodes within the same broadcast domain (for Ethernet and Wi-Fi) or ring (for token ring or FDDI). Computers attached to the same network hub satisfy this requirement, which is why network switches are used to combat malicious use of promiscuous mode. A router may monitor all traffic that it routes.

Promiscuous mode is often used to diagnose network connectivity issues. There are programs that make use of this feature to show the user all the data being transferred over the network. Some protocols like FTP and Telnet transfer data and passwords in clear text, without encryption, and network scanners can see this data. Therefore, computer users are encouraged to stay away from insecure protocols like telnet and use more secure ones such as SSH.

Detection

As promiscuous mode can be used in a malicious way to *sniff* on a network, one might be interested in detecting network devices that are in promiscuous mode. In promiscuous mode, some software might send responses to frames even though they were addressed to another machine. However, experienced sniffers can prevent this (e.g., using carefully designed firewall settings). An example is sending a ping (ICMP echo request) with the wrong MAC address but the right IP address. If your firewall blocks all ICMP traffic, this will be prevented.

Some applications that use promiscuous mode

- NetScout Sniffer
- OmniPeek
- Capsa for WiFi
- Aircrack-ng
- KisMAC (used for WLAN)
- AirSnort (used for WLAN)
- Wireshark (formerly *Ethereal*)
- tcpdump
- IPTraf
- PRTG
- Kismet
- VMware's VMnet Bridging (networking)
- Cain and Abel
- Driftnet Software
- Microsoft Windows Network Bridge
- XLink Kai

- WC3Banlist
- Snort
- ntop
- Firesheep
- VirtualBox (bridge networking mode)
- CommView for WiFi
- AccessData SilentRunner

External links

SearchSecurity.com definition of promiscuous mode [1]

References

[1] http://searchsecurity.techtarget.com/definition/promiscuous-mode

Protocol data unit

In telecommunications, the term **protocol data unit** (**PDU**) has the following meanings:

1. Information that is delivered as a unit among peer entities of a network and that may contain control information, address information, or data.
2. In a layered system, a unit of data which is specified in a protocol of a given layer and which consists of protocol-control information and possibly user data of that layer. For example: Bridge PDU or iSCSI PDU[1]

PDUs are relevant in relation to each of the first 4 layers of the OSI model as follows:

1. The Layer 1 (Physical Layer) PDU is the bit
2. The Layer 2 (Data Link Layer) PDU is the frame
3. The Layer 3 (Network Layer) PDU is the packet
4. The Layer 4 (Transport Layer) PDU is the segment (e.g. TCP segment)

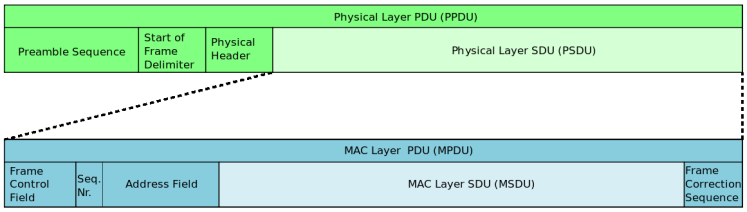
(Layer 5 and above are referred to as data.)

Given a context pertaining to a specific layer, PDU is sometimes used as a synonym for its representation at that layer.

Packet-switched data networks

In the context of packet-switched data networks, a protocol data unit (PDU) is best understood in relation to a service data unit (SDU). The features or services of the network are implemented in distinct "layers". For example, sending ones and zeros across a wire, fiber, etc. is done by the physical layer, organizing the ones and zeros into chunks of data and getting them safely to the right place on the wire is done by the data link layer, passing data chunks over multiple connected networks is done by the network layer and delivery of the data to the right software application at the destination is done by the transport layer. Between the layers (and between the application and the top-most layer), the layers pass service data units across the interfaces. The application or higher layer understands the structure of the data in the SDU, but the lower layer at the interface does not; it treats it as payload, undertaking to get it to the same interface at the destination. In order to do this, the protocol layer will add to the SDU certain data it needs to perform its function. For example, it might add a port number to identify the application, a network address to help with routing, a code to identify the type of data in the packet and error-checking information. All this additional information, plus the original service data unit from the higher layer, constitutes the *protocol data unit* at this layer. The significance of this is that the PDU is the structured information that is passed to a matching protocol

layer further along on the data's journey that allows the layer to deliver its intended function or service. The matching layer, or "peer", decodes the data to extract the original service data unit, decide if it is error-free and where to send it next, etc. Unless we have already arrived at the lowest (physical) layer, the PDU is passed to the peer using services of the next lower layer in the protocol "stack". When the PDU passes over the interface from the layer that constructed it to the layer that merely delivers it (and therefore does not understand its internal structure), it becomes a service data unit to that layer. The addition of addressing and control information (which is called encapsulation) to an SDU to form a PDU and the passing of that PDU to the next lower layer as an SDU repeats until the lowest layer is reached and the data passes over some medium as a physical signal.



MAC layer PDU becomes physical layer SDU

The above process can be likened to the mail system in which a letter (SDU) is placed in an envelope on which is written an address (addressing and control information) making it a PDU. The sending post office might look only at the post code and place the letter in a mail bag so that the address on the envelope can no longer be seen, making it now an SDU. The mail bag is labelled with the destination post code and so becomes a PDU, until it is combined with other bags in a crate, when it is now an SDU, and the crate is labelled with the region to which all the bags are to be sent, making the crate a PDU. When the crate reaches the destination matching its label, it is opened and the bags (SDUs) removed only to become PDUs when someone reads the code of the destination post office. The letters themselves are SDUs when the bags are opened but become PDUs when the address is read for final delivery. When the addressee finally opens the envelope, the top-level SDU, the letter itself, emerges.

References

[1] *This article incorporates public domain material from the General Services Administration* document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm).

External links

- comp.protocols.iso FAQ (http://www.uni-giessen.de/faq/archiv/osi-protocols/msg00000.html) (search for "PDU")

Proxy ARP

Proxy ARP (Address Resolution Protocol) is a technique by which a device on a given network answers the ARP queries for a network address that is not on that network. The ARP Proxy is aware of the location of the traffic's destination, and offers its own MAC address in reply, effectively saying, "send it to me, and I'll get it to where it needs to go." Serving as an ARP Proxy for another host effectively directs LAN traffic to the Proxy. The "captured" traffic is then typically routed by the Proxy to the intended destination via another interface or via a tunnel.

The process which results in the node responding with its own MAC address to an ARP request for a different IP address for proxying purposes is sometimes referred to as 'publishing'. A Proxy ARP Simulation [1] visualizes how a router responds to ARP request on behalf of the target host at different networks. See ARP cache updates at host, router, remote host.

Uses

Below are some typical uses for proxy ARP:

Joining a broadcast LAN with serial links (e.g., dialup or VPN connections).

> Assume an Ethernet broadcast domain (e.g., a group of stations connected to the same hub) using a certain IPv4 address range (e.g., 192.168.0.0/24, where 192.168.0.1 – 192.168.0.127 are assigned to wired nodes). One or more of the nodes is an access router accepting dialup or VPN connections. The access router gives the dial-up nodes IP addresses in the range 192.168.0.128 – 192.168.0.254; for this example, assume a dial-up node gets IP address 192.168.0.254.
>
> The access router uses Proxy ARP to make the dial-up node present in the subnet without being wired into the Ethernet: the access server 'publishes' its own MAC address for 192.168.0.254. Now, when another node wired into the Ethernet wants to talk to the dial-up node, it will ask on the network for the MAC address of 192.168.0.254 and find the access server's MAC address. It will therefore send its IP packets to the access server, and the access server will know to pass them on to the particular dial-up node. All dial-up nodes therefore appear to the wired Ethernet nodes as if they are wired into the same Ethernet subnet.

Taking multiple addresses from a LAN

> Assume a station (e.g., a server) with an interface (10.0.0.2) connected to a network (10.0.0.0/24). Certain applications may require multiple IP addresses on the server. Provided the addresses have to be from the 10.0.0.0/24 range, the way the problem is solved is through Proxy ARP. Additional addresses (say, 10.0.0.230-10.0.0.240) are aliased to the loopback interface of the server (or assigned to special interfaces, the latter typically being the case with VMware/UML/jails/vservers/other virtual server environments) and 'published' on the 10.0.0.2 interface (although many operating systems allow direct allocation of multiple addresses to one interface, thus eliminating the need for such tricks).

On a firewall

> In this scenario a firewall can be configured with a single IP address. One simple example of a use for this would be placing a firewall in front of a single host or group of hosts on a subnet. Example- A network (10.0.0.0/8) has a server which should be protected (10.0.0.20) a proxy-arp firewall can be placed in front of the server. In this way the server is put behind a firewall without making any changes to the network at all.

Mobile-IP

> In case of Mobile-IP the Home Agent uses Proxy ARP in order to receive messages on behalf of the Mobile Node, so that it can forward the appropriate message to the actual mobile node's address (Care-of address).

Transparent subnet gatewaying

A setup that involves two physical segments sharing the same IP subnet and connected together via a router. This use is documented in RFC 1027.

Advantages

The advantage of Proxy ARP over other networking schemes is simplicity. A network can be extended using this technique without the knowledge of the upstream router.

For example, suppose a host, say A, wants to contact another host B, where B is on a different subnet/broadcast domain than A. For this, host A will send an ARP request with a Destination IP address of B in its ARP packet. The multi-homed router which is connected to both the subnets, responds to host A's request with its MAC address instead of host B's actual MAC address, thus proxying for host B. In the due course of time, when host A sends a packet to the router which is actually destined to host B, the router just forwards the packet to host B. The communication between host A and B is totally unaware of the router proxying for each other.

Disadvantages

Disadvantage of Proxy ARP include scalability (ARP resolution is required for every device routed in this manner) and reliability (no fallback mechanism is present, and masquerading can be confusing in some environments). ARP manipulation techniques, however, are the basis for protocols providing redundancy on broadcast networks (e.g., Ethernet), most notably CARP and Virtual Router Redundancy Protocol.

Proxy ARP can create DoS attacks on networks if misconfigured. For example a misconfigured router with proxy ARP has the ability to receive packets destined for other hosts (as it gives its own MAC address in response to ARP requests for other hosts/routers), but may not have the ability to correctly forward these packets on to their final destination, thus blackholing the traffic.

Further reading

- RFC 925 – Multi-LAN Address Resolution
- RFC 1027 – Using ARP to Implement Transparent Subnet Gateways
- W. Richard Stevens. The Protocols (TCP/IP Illustrated, Volume 1). Addison-Wesley Professional; 1st edition (December 31, 1993). ISBN 0-201-63346-9

References

[1] http://pre.visualland.net/view.php?cid=862&protocol=ARP&title=6.%20ARP-Router%20proxy

Quality of service

The **quality of service** (**QoS**) refers to several related aspects of telephony and computer networks that allow the transport of traffic with special requirements. In particular, much technology has been developed to allow computer networks to become as useful as telephone networks for audio conversations, as well as supporting new applications with even more strict service demands.

Definitions

In the field of telephony, quality of service was defined by the ITU in 1994.[1] Quality of service comprises requirements on all the aspects of a connection, such as service response time, loss, signal-to-noise ratio, cross-talk, echo, interrupts, frequency response, loudness levels, and so on. A subset of telephony QoS is grade of service (GoS) requirements, which comprises aspects of a connection relating to capacity and coverage of a network, for example guaranteed maximum blocking probability and outage probability.[2]

In the field of computer networking and other packet-switched telecommunication networks, the traffic engineering term refers to resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over IP, online games and IP-TV, since these often require fixed bit rate and are delay sensitive, and in networks where the capacity is a limited resource, for example in cellular data communication.

A network or protocol that supports QoS may agree on a traffic contract with the application software and reserve capacity in the network nodes, for example during a session establishment phase. During the session it may monitor the achieved level of performance, for example the data rate and delay, and dynamically control scheduling priorities in the network nodes. It may release the reserved capacity during a tear down phase.

A best-effort network or service does not support quality of service. An alternative to complex QoS control mechanisms is to provide high quality communication over a best-effort network by over-provisioning the capacity so that it is sufficient for the expected peak traffic load. The resulting absence of network congestion eliminates the need for QoS mechanisms.

QoS is sometimes used as a quality measure, with many alternative definitions, rather than referring to the ability to reserve resources. Quality of service sometimes refers to the level of quality of service, i.e. the guaranteed service quality. High QoS is often confused with a high level of performance or achieved service quality, for example high bit rate, low latency and low bit error probability.

An alternative and disputable definition of QoS, used especially in application layer services such as telephony and streaming video, is requirements on a metric that reflects or predicts the subjectively experienced quality. In this context, QoS is the acceptable cumulative effect on subscriber satisfaction of all imperfections affecting the service. Other terms with similar meaning are the quality of experience (QoE) subjective business concept, the required “user perceived performance”,[3] the required “degree of satisfaction of the user” or the targeted “number of happy customers”. Examples of measures and measurement methods are Mean Opinion Score (MOS), Perceptual Speech Quality Measure (PSQM) and Perceptual Evaluation of Video Quality (PEVQ). See also subjective video quality.

History

Conventional Internet routers and LAN switches operate on a best effort basis. This equipment is less expensive, less complex and faster and thus more popular than competing more complex technologies that provided QoS mechanisms. There were four “Type of service” bits and three “Precedence” bits provided in each IP packet header, but they were not generally respected. These bits were later re-defined as DiffServ Code Points (DSCP) and are sometimes honored in peered links on the modern Internet.

With the advent of IPTV and IP telephony, QoS mechanisms are increasingly available to the end user.

A number of attempts for layer 2 technologies that add QoS tags to the data have gained popularity during the years, but then lost attention. Examples are Frame relay and ATM. Recently, MPLS (a technique between layer 2 and 3) have gained some attention. However, today Ethernet may offer QoS through its 802.1p. Ethernet is, by far, the most popular layer 2 technology.

In Ethernet, Virtual LANs (VLAN) may be used to separate different QoS levels. For example in fibre-to-the-home switches typically offer several Ethernet ports connected to different VLAN:s. One VLAN may be used for Internet access (low priority), one for IPTV (higher priority) and one for IP telephony (highest priority). Different Internet service providers may use the different VLANs.

Qualities of traffic

In packet-switched networks, quality of service is affected by various factors, which can be divided into “human” and “technical” factors. Human factors include: stability of service, availability of service, delays, user information. Technical factors include: reliability, scalability, effectiveness, maintainability, grade of service, etc.[4]

Many things can happen to packets as they travel from origin to destination, resulting in the following problems as seen from the point of view of the sender and receiver:

Low throughput

Due to varying load from other users sharing the same network resources, the bit rate (the maximum throughput) that can be provided to a certain data stream may be too low for realtime multimedia services if all data streams get the same scheduling priority.

Dropped packets

The routers might fail to deliver (*drop*) some packets if their data is corrupted or they arrive when their buffers are already full. The receiving application may ask for this information to be retransmitted, possibly causing severe delays in the overall transmission.

Errors

Sometimes packets are corrupted due to bit errors caused by noise and interference, especially in wireless communications and long copper wires. The receiver has to detect this and, just as if the packet was dropped, may ask for this information to be retransmitted.

Latency

It might take a long time for each packet to reach its destination, because it gets held up in long queues, or takes a less direct route to avoid congestion. This is different from throughput, as the delay can build up over time, even if the throughput is almost normal. In some cases, excessive latency can render an application such as VoIP or online gaming unusable.

Jitter

Packets from the source will reach the destination with different delays. A packet's delay varies with its position in the queues of the routers along the path between source and destination and this position can vary unpredictably. This variation in delay is known as jitter and can seriously affect the quality of streaming audio

and/or video.

Out-of-order delivery

When a collection of related packets is routed through a network, different packets may take different routes, each resulting in a different delay. The result is that the packets arrive in a different order than they were sent. This problem requires special additional protocols responsible for rearranging out-of-order packets to an isochronous state once they reach their destination. This is especially important for video and VoIP streams where quality is dramatically affected by both latency and lack of sequence.

Applications

A defined quality of service may be desired or required for certain types of network traffic, for example:

- Streaming media specifically
  - Internet protocol television (IPTV)
  - Audio over Ethernet
  - Audio over IP
- IP telephony also known as Voice over IP (VoIP)
- Videoconferencing
- Telepresence
- Circuit Emulation Service
- Safety-critical applications such as remote surgery where availability issues can be hazardous
- Network operations support systems either for the network itself, or for customers' business critical needs
- Online games where real-time lag can be a factor
- Industrial control systems protocols such as Ethernet/IP which are used for real-time control of machinery

These types of service are called *inelastic*, meaning that they require a certain minimum level of bandwidth and a certain maximum latency to function. By contrast, *elastic* applications can take advantage of however much or little bandwidth is available. Bulk file transfer applications that rely on TCP are generally elastic.

Mechanisms

In circuit switched networks, especially those intended for voice transmission, such as Asynchronous Transfer Mode (ATM) or GSM, have QoS in the core protocol and do not need additional procedures to achieve it. Shorter data units and built-in QoS were some of the unique selling points of ATM for applications such as video on demand.

When the expense of mechanisms to provide QoS is justified, network customers and providers can enter into a contractual agreement termed a service level agreement (SLA) which specifies guarantees for the ability of a network/protocol to give guaranteed performance/throughput/latency bounds based on mutually agreed measures, usually by prioritizing traffic. In other approaches, resources are reserved at each step on the network for the call as it is set up.

Over-provisioning

An alternative to complex QoS control mechanisms is to provide high quality communication by generously over-provisioning a network so that capacity is based on peak traffic load estimates. This approach is simple and economical for networks with predictable and light traffic loads. The performance is reasonable for many applications. This might include demanding applications that can compensate for variations in bandwidth and delay with large receive buffers, which is often possible for example in video streaming. Over-provisioning can be of limited use, however, in the face of transport protocols (such as TCP) that over time exponentially increase the amount of data placed on the network until all available bandwidth is consumed and packets are dropped. Such greedy protocols tend to increase latency and packet loss for all users.

Commercial VoIP services are often competitive with traditional telephone service in terms of call quality even though QoS mechanisms are usually not in use on the user's connection to his ISP and the VoIP provider's connection to a different ISP. Under high load conditions, however, VoIP may degrade to cell-phone quality or worse. The mathematics of packet traffic indicate that network requires just 60% more raw capacity under conservative assumptions.[5]

The amount of over-provisioning in interior links required to replace QoS depends on the number of users and their traffic demands. This limits usability of over-provisioning. Newer more bandwidth intensive applications and the addition of more users results in the loss of over-provisioned networks. This then requires a physical update of the relevant network links which is an expensive process. Thus over-provisioning cannot be blindly assumed on the Internet.

IP and Ethernet efforts

Unlike single-owner networks, the Internet is a series of exchange points interconnecting private networks.[6] Hence the Internet's core is owned and managed by a number of different network service providers, not a single entity. Its behavior is much more stochastic or unpredictable. Therefore, research continues on QoS procedures that are deployable in large, diverse networks.

There are two principal approaches to QoS in modern packet-switched IP networks, a parameterized system based on an exchange of application requirements with the network, and a prioritized system where each packet identifies a desired service level to the network.

- Integrated services (“IntServ”) implements the parameterized approach. In this model, applications use the Resource Reservation Protocol (RSVP) to request and reserve resources through a network.
- Differentiated services (“DiffServ”) implements the prioritized model. DiffServ marks packets according to the type of service they desire. In response to these markings, routers and switches use various queueing strategies to tailor performance to expectations. DiffServ Code Point (DSCP) markings use the first 6 bits in the ToS field of the IP(v4) packet header.

Early work used the integrated services (IntServ) philosophy of reserving network resources. In this model, applications used the Resource reservation protocol (RSVP) to request and reserve resources through a network. While IntServ mechanisms do work, it was realized that in a broadband network typical of a larger service provider, Core routers would be required to accept, maintain, and tear down thousands or possibly tens of thousands of reservations. It was believed that this approach would not scale with the growth of the Internet, and in any event was antithetical to the notion of designing networks so that Core routers do little more than simply switch packets at the highest possible rates.

The second and currently accepted approach is differentiated services (DiffServ). In the DiffServ model, packets are marked according to the type of service they need. In response to these markings, routers and switches use various queuing strategies to tailor performance to requirements. — At the IP layer, differentiated services code point (DSCP) markings use the 6 bits in the IP packet header. At the MAC layer, VLAN IEEE 802.1Q and IEEE 802.1p can be used to carry essentially the same information.

Routers supporting DiffServ use multiple queues for packets awaiting transmission from bandwidth constrained (e.g., wide area) interfaces. Router vendors provide different capabilities for configuring this behavior, to include the number of queues supported, the relative priorities of queues, and bandwidth reserved for each queue.

In practice, when a packet must be forwarded from an interface with queuing, packets requiring low jitter (e.g., VoIP or videoconferencing) are given priority over packets in other queues. Typically, some bandwidth is allocated by default to network control packets (such as Internet Control Message Protocol and routing protocols), while best effort traffic might simply be given whatever bandwidth is left over.

At the Media Access Control (MAC) layer, VLAN IEEE 802.1Q and IEEE 802.1p can be used to carry essentially the same information as used by DiffServ.

Cisco IOS NetFlow and the Cisco Class Based QoS (CBQoS) Management Information Base (MIB) are marketed by Cisco systems. [7]

Additional bandwidth management mechanisms may be used to further engineer performance, to include:

- Traffic shaping (rate limiting):
  - Token bucket
  - Leaky bucket
  - TCP rate control—artificially adjusting TCP window size as well as controlling the rate of ACKs being returned to the sender
- Scheduling algorithms:
  - Weighted fair queuing (WFQ)
  - Class based weighted fair queuing
  - Weighted round robin (WRR)
  - Deficit weighted round robin (DWRR)
  - Hierarchical Fair Service Curve (HFSC)
- Congestion avoidance:
  - RED, WRED — Lessens the possibility of port queue buffer tail-drops and this lowers the likelihood of TCP global synchronization
  - Policing (marking/dropping the packet in excess of the committed traffic rate and burst size)
  - Explicit congestion notification
  - Buffer tuning

While DiffServ is used in many sophisticated enterprise networks, it has not been widely deployed in the Internet. Internet peering arrangements are already complex, and there appears to be no enthusiasm among providers for supporting QoS across peering connections, or agreement about what policies should be supported in order to do so.

One compelling example of the need for QoS on the Internet relates to congestion collapse. The Internet relies on congestion avoidance protocols, as built into Transmission Control Protocol (TCP), to reduce traffic under conditions that would otherwise lead to "meltdown". QoS applications such as VoIP and IPTV, because they require largely constant bitrates and low latency cannot use TCP and cannot otherwise reduce their traffic rate to help preventcongestion. QoS contracts limit traffic that can be offered to the Internet and thereby enforce traffic shaping that can prevent it from becoming overloaded, hence they're an indispensable part of the Internet's ability to handle a mix of real-time and non-real-time traffic without meltdown.

Protocols

- The Type of Service (ToS) field in the IP(v4) header (now superseded by DiffServ)
- IP Differentiated services (DiffServ)
- IP Integrated services (IntServ)
- Resource reSerVation Protocol (RSVP)
- Multiprotocol Label Switching (MPLS) provides eight QoS classes
- RSVP-TE
- Frame relay
- X.25
- Some ADSL modems
- Asynchronous Transfer Mode (ATM)
- IEEE 802.1p

- IEEE 802.1Q
- IEEE 802.11e
- HomePNA Home networking over coax and phone wires
- The ITU-T G.hn standard provides QoS by means of “Contention-Free Transmission Opportunities” (CFTXOPs) which are allocated to flows which require QoS and which have negotiated a “contract” with the network controller. G.hn also supports non-QoS operation by means of “Contention-based Time Slots”.
- Audio Video Bridging

End-to-end quality of service

End-to-end quality of service can require a method of coordinating resource allocation between one autonomous system and another. The Internet Engineering Task Force (IETF) defined the Resource Reservation Protocol (RSVP) for bandwidth reservation, as a proposed standard in 1997.[8] RSVP is an end-to-end bandwidth reservation protocol. The traffic engineering version, RSVP-TE, is used in many networks to establish traffic-engineered Multiprotocol Label Switching (MPLS) label-switched paths. The IETF also defined Next Steps in Signaling (NSIS)[9] with QoS signalling as a target. NSIS is a development and simplification of RSVP.

Research consortia such as "end-to-end quality of service support over heterogeneous networks" (EuQoS, from 2004 through 2007)[10] and fora such as the IPsphere Forum[11] developed more mechanisms for handshaking QoS invocation from one domain to the next. IPsphere defined the Service Structuring Stratum (SSS) signaling bus in order to establish, invoke and (attempt to) assure network services. EuQoS conducted experiments to integrate Session Initiation Protocol, Next Steps in Signaling and IPsphere's SSS with an estimated cost of about 15.6 million Euro and published a book.[12] [13]

A research project Multi Service Access Everywhere (MUSE) defined another QoS concept in a first phase from January 2004 through February 2006, and a second phase from January 2006 through 2007.[14] [15] [16] Another research project named PlaNetS was proposed for European funding circa 2005.[17] A broader European project called "Architecture and design for the future Internet" known as 4WARD had a budgest estimated at 23.4 million Euro and was funded from January 2008 through June 2010.[18] It included a "Quality of Service Theme" and published a book.[19] [20]

Circumvention

Strong cryptography network protocols such as Secure Sockets Layer, I2P, and virtual private networks obscure the data transferred using them. As all electronic commerce on the Internet requires the use of such strong cryptography protocols, unilaterally downgrading the performance of encrypted traffic creates an unacceptable hazard for customers. Yet, encrypted traffic is otherwise unable to undergo deep packet inspection for QoS.

Doubts about quality of service over IP

The Internet2 project found, in 2001, that the QoS protocols were probably not deployable inside its Abilene Network with equipment available at that time.[21] Equipment available at the time relied on software to implement QoS. The group also predicted that “logistical, financial, and organizational barriers will block the way toward any bandwidth guarantees” by protocol modifications aimed at QoS. [22] They believed that the economics would encourage network providers to deliberately erode the quality of best effort traffic as a way to push customers to higher priced QoS services. Instead they proposed over-provisioning of capacity as more cost-effective at the time.[21] [22]

The Abilene network study was the basis for the testimony of Gary Bachula to the US Senate Commerce Committee's hearing on Network Neutrality in early 2006. He expressed the opinion that adding more bandwidth was more effective than any of the various schemes for accomplishing QoS they examined.[23]

Bachula's testimony has been cited by proponents of a law banning quality of service as proof that no legitimate purpose is served by such an offering. This argument is dependent on the assumption that over-provisioning isn't a form of QoS and that it is always possible. Cost and other factors affect the ability of carriers to build and maintain permanently over-provisioned networks

Mobile (cellular) QoS

Mobile cellular service providers may offer mobile QoS to customers just as the fixed line PSTN services providers and Internet Service Provides (ISP) may offer QoS. QoS mechanisms are always provided for circuit switched services, and are essential for non-elastic services, for example streaming multimedia.

Mobility adds complication to the QoS mechanisms, for several reasons:

- A phone call or other session may be interrupted after a handover, if the new base station is overloaded. Unpredictable handovers make it impossible to give an absolute QoS guarantee during a session initiation phase.
- The pricing structure is often based on per-minute or per-megabyte fee rather than flat rate, and may be different for different content services.
- A crucial part of QoS in mobile communications is Grade of Service, involving outage probability (the probability that the mobile station is outside the service coverage area, or affected by co-channel interference, i.e. crosstalk) blocking probability (the probability that the required level of QoS can not be offered) and scheduling starvation. These performance measures are affected by mechanisms such as mobility management, radio resource management, admission control, fair scheduling, channel-dependent scheduling etc.

Standards

Quality of service in the field of telephony, was first defined in 1994 in the ITU-T Recommendation E.800. This definition is very broad, listing 6 primary components: Support, Operability, Accessibility, Retainability, Integrity and Security.[1] A 1995 recommendation X.902 included a definition is the OSI reference model.[24] In 1998 the ITU published a document discussing QoS in the field of data networking. X.641 offers a means of developing or enhancing standards related to QoS and provide concepts and terminology that will assist in maintaining the consistency of related standards.[25]

The main QoS-related IETF Request For Comments (RFC)s are Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers [26] (RFC 2474), and Resource ReSerVation Protocol (RSVP) [27] (RFC 2205); both these are discussed above. The IETF has also published two RFCs giving background on QoS: RFC 2990: Next Steps for the IP QoS Architecture [28], and RFC 3714: IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet [29].

Open source software

- Linux Advanced Routing & Traffic Control (from 2000 to 2005)[30]
- Linux Bandwidth Arbitrator (2003 through 2005)[31]
- Zero Shell[32]
- mod_qos adding QoS to the Apache HTTP Server

References

[1] "E.800: Terms and definitions related to quality of service and network performance including dependability" (http://www.itu.int/rec/T-REC-E.800/en). *ITU-T Recommendation*. August 1994. . Retrieved October 14, 2011. Updated September 2008 as *Definitions of terms related to quality of service*

[2] Teletraffic Engineering Handbook (http://www.com.dtu.dk/teletraffic/handbook/telenook.pdf) ITU-T Study Group 2 (350 pages, 4·48MiB)(It uses abbreviation GoS instead of QoS)

[3] Leonard Franken. Quality of Service Management: A Model-Based Approach. PhD thesis, Centre for Telematics and Information Technology, 1996.
[4] Peuhkuri M., IP Quality of Service, Helsinki University of Technology, Laboratory of Telecommunications Technology, 1999.
[5] Yuksel, M.; Ramakrishnan, K. K.; Kalyanaraman, S.; Houle, J. D.; Sadhvani, R. (2007). "IEEE International Workshop on Quality of Service (IWQoS'07)" (http://iwqos07.ece.ucdavis.edu/) (PDF). Evanston, IL, USA. pp. 109–112. doi:10.1109/IWQOS.2007.376555. .
[6] An Evening With Robert Kahn (http://archive.computerhistory.org/lectures/an_eveninig_with_robert_kahn.lecture.2007.01.09.wmv), from Computer History Museum, 9 Jan 2007
[7] Ben Erwin (December 16, 2008). "How To Manage QoS In Your Environment, Part 1 of 3" (http://www.networkperformancedaily.com/2008/12/whiteboard_series_how_to_manag.html). *Network Performance Daily video*. NetQoS. . Retrieved Ocotber 15, 2011.
[8] Bob Braden ed. L. Zhang, S. Berson, S. Herzog, S. Jamin (September 1997). "Resource ReSerVation Protocol (RSVP)" (http://tools.ietf.org/html/rfc2205). *RFC 2205*. IETF. . Retrieved October 12, 2011.
[9] "Next Steps in Signaling" Charter (http://datatracker.ietf.org/wg/nsis/charter/)
[10] "EuQoS - End-to-end Quality of Service support over heterogeneous networks" (http://web.archive.org/web/20070430123408/http://www.euqos.eu/). *Project website*. 2004–2006. Archived from the original (http://www.euqos.eu/) on April 30, 2007. . Retrieved October 12, 2011.
[11] IPSphere: Enabling Advanced Service Delivery (http://www.tmforum.org/ipsphere/)
[12] "End-to-end quality of service support over heterogeneous networks" (http://cordis.europa.eu/search/index.cfm?fuseaction=proj.document&PJ_LANG=EN&PJ_RCN=6903468). *Project description*. European Community Research and Development Information Service. . Retrieved October 12, 2011.
[13] Torsten Braun; Thomas Staub (2008). *End-to-end quality of service over heterogeneous networks* (http://books.google.com/books?id=ajnpzZOOVUgC&pg=PA161). Springer. ISBN 978-3-540-79119-5. .
[14] "Multi Service Access Everywhere (MUSE)" (http://www.ist-muse.org/). *Project website*. . Retrieved October 12, 2011.
[15] "Multi Service Access Everywhere" (http://cordis.europa.eu/search/index.cfm?fuseaction=proj.document&PJ_LANG=EN&PJ_RCN=12052357). *Project description*. European Community Research and Development Information Service. . Retrieved October 12, 2011.
[16] "Multi Service Access Everywhere" (http://cordis.europa.eu/search/index.cfm?fuseaction=proj.document&PJ_LANG=EN&PJ_RCN=9151777). *Project description*. European Community Research and Development Information Service. . Retrieved October 12, 2011.
[17] "PlaNetS QoS Solution" (http://web.archive.org/web/20091112202614/http://www.medea-planets.eu/QoSsolution.php?). *Project website*. Archived from the original (http://www.medea-planets.eu/QoSsolution.php) on November 12, 2009. . Retrieved October 12, 2011.
[18] "4WARD: Architecture and design for the future Internet" (http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=85316). *Project description*. European Community Research and Development Information Service. . Retrieved October 15, 2011.
[19] "Going 4WARD" (http://www.4ward-project.eu/index.php?s=file_download&id=90) (PDF). *Project newsletter*. June 2010. . Retrieved October 15, 2011.
[20] Luís M. Correia; Joao Schwarz (FRW) da Silva (January 30, 2011). *Architecture and Design for the Future Internet: 4WARD EU Project* (http://books.google.com/books?id=DbucHQnOMW8C). Springer. ISBN 978-90-481-9345-5. .
[21] Benjamin Teitelbaum, Stanislav Shalunov (May 3, 2002). "Why Premium IP Service Has Not Deployed (and Probably Never Will)" (http://qbone.internet2.edu/papers/non-architectural-problems.txt). *Draft Informational Document*. Internet2 QoS Working Group. Archived (http://www.webcitation.org/5shCiXna8) from the original on September 12, 2010. . Retrieved October 15, 2011.
[22] Andy Oram (June 11, 2002). "A Nice Way to Get Network Quality of Service?" (http://www.oreillynet.com/pub/a/network/2002/06/11/platform.html). *Platform Independent column*. O'Reilly. Archived (http://www.webcitation.org/5shCWW1GT) from the original on September 12, 2010. . Retrieved October 15, 2011.
[23] Gary Bachula (February 7, 2006). "Testimony of Gary R. Bachula, Vice President, Internet2" (http://commerce.senate.gov/pdf/bachula-020706.pdf). pp. 2–3. . Retrieved October 15, 2011.
[24] "X.902:Information technology – Open Distributed Processing – Reference model: Foundations" (http://www.itu.int/rec/T-REC-X.902/en). *ITU-T Recommendation*. November 1995. . Retrieved October 14, 2011. Updated October 2009.
[25] "X.641: Information technology - Quality of service: framework" (http://www.itu.int/rec/T-REC-X.641/en). *ITU-T Recommendation*. December 1997. .
[26] http://www.apps.ietf.org/rfc/rfc2474.html
[27] http://www.apps.ietf.org/rfc/rfc2205.html
[28] http://www.apps.ietf.org/rfc/rfc2990.html
[29] http://www.apps.ietf.org/rfc/rfc3714.html
[30] "Advanced Routing & Traffic Control HOWTO" (http://lartc.org/). August 21, 2005. . Retrieved October 14, 2011.
[31] "Linux Bandwidth Arbitrator" (http://www.bandwidtharbitrator.com/). APConnections. . Retrieved October 14, 2011.
[32] Fulvio Ricciardi. "QoS and Traffic Shaping in Transparent Bridge mode" (http://www.zeroshell.net/eng/qos/). *Router/Bridge Linux Firewall website*. ZeroShell Net Services. . Retrieved October 15, 2011.

Further reading

- *Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective* by Xipeng Xiao (Morgan Kaufmann, 2008, ISBN 0-12-373693-5)
- *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice* by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, ISBN 0-12-370549-5)
- Lelli, F. Maron, G. Orlando, S. Client Side Estimation of a Remote Service Execution (http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=4674430). 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07.
- *QoS Over Heterogeneous Networks* by Mario Marchese (Wiley, 2007, ISBN 978-0-470-01752-4)
- XiPeng Xiao (September 8, 2008). *Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective* (http://books.google.com/books?id=oVlspevE_78C+). Morgan Kaufmann. ISBN 978-0-12-373693-2.
- RFC 1633: Integrated Services in the Internet Architecture: an Overview (http://www.apps.ietf.org/rfc/rfc1633.html)
- RFC 2475: An Architecture for Differentiated Services (http://www.apps.ietf.org/rfc/rfc2475.html)
- RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels (http://www.apps.ietf.org/rfc/rfc3209.html)

External links

- Nate Hoy. "Implementing QoS" (http://vonage.nmhoy.net/qos.html). *Vonage Forum*. Retrieved October 14, 2011.
- Cisco's Internetworking Technology Handbook (http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook#Quality_of_Service_Networking)
- Henning Schulzrinne (January 9, 2008). "Network Quality of Service" (http://www.cs.columbia.edu/~hgs/internet/qos.html). *Columbia University faculty website*. Retrieved October 14, 2011.
- "Quality of Service" (http://technet.microsoft.com/en-us/network/bb530836.aspx). *Microsoft TechNet*. March 31, 2011. Retrieved October 14, 2011.

Reverse Address Resolution Protocol

The **Reverse Address Resolution Protocol** (**RARP**) is an obsolete computer networking protocol used by a host computer to request its Internet Protocol (IPv4) address from an administrative host, when it has available its Link Layer or hardware address, such as a MAC address.

RARP is described in Internet Engineering Task Force (IETF) publication RFC 903.[1] It has been rendered obsolete by the Bootstrap Protocol (BOOTP) and the modern Dynamic Host Configuration Protocol (DHCP), which both support a much greater feature set than RARP.

RARP requires one or more server hosts to maintain a database of mappings of Link Layer addresses to their respective protocol addresses. Media Access Control (MAC) addresses needed to be individually configured on the servers by an administrator. RARP was limited to serving only IP addresses.

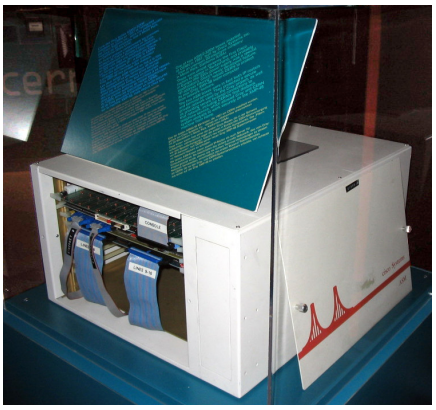
Reverse ARP differs from the Inverse Address Resolution Protocol (InARP) described in RFC 2390, which is designed to obtain the IP address associated with another host's MAC address. InARP is the complement of the Address Resolution Protocol used for the reverse lookup.

References

[1] RFC 903, *A Reverse Address Resolution Protocol*, R. Finlayson, T. Mann, J. Mogul, M. Theimer (June 1984)

Router (computing)

A **router** is a device that forwards data packets between computer networks, creating an overlay internetwork. A router is connected to two or more data lines from different networks. When a data packet comes in on one of the lines, the router reads the address information in the packet to determine its ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey. Routers perform the "traffic directing" functions on the Internet. A data packet is typically forwarded from one router to another through the networks that constitute the internetwork until it gets to its destination node.[1]



A Cisco ASM/2-32EM router deployed at CERN in 1987

The most familiar type of routers are home and small office routers that simply pass data, such as web pages and email, between the home computers and the owner's cable or DSL modem, which connects to the Internet (ISP). However more sophisticated routers range from enterprise routers, which connect large business or ISP networks up to the powerful core routers that forward data at high speed along the optical fiber lines of the Internet backbone.

Juniper SRX210 service gateway router

Applications

When multiple routers are used in interconnected networks, the routers exchange information about destination addresses, using a dynamic routing protocol. Each router builds up a table listing the preferred routes between any two systems on the interconnected networks. A router has interfaces for different physical types of network connections, (such as copper cables, fiber optic, or wireless transmission). It also contains firmware for different networking protocol standards. Each network interface uses this specialized computer software to enable data packets to be forwarded from one protocol transmission system to another.

Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different sub-network address. The subnets addresses recorded in the router do not necessarily map directly to the physical interface connections.[2] A router has two stages of operation called planes:[3]

- Control plane: A router records a routing table listing what route should be used to forward a data packet, and through which physical interface connection. It does this using internal pre-configured addresses, called static routes.
- Forwarding plane: The router forwards data packets between incoming and outgoing interface connections. It routes it to the correct network type using information that the packet header contains. It uses data recorded in the routing table control plane.

A typical home or small office router showing the ADSL telephone line and Ethernet network cable connections

Routers may provide connectivity within enterprises, between enterprises and the Internet, and between internet service providers (ISPs) networks. The largest routers (such as the Cisco CRS-1 or Juniper T1600) interconnect the various ISPs, or may be used in large enterprise networks.[4] Smaller routers usually provide connectivity for typical home and office networks. Other networking solutions may be provided by a backbone Wireless Distribution System (WDS), which avoids the costs of introducing networking cables into buildings.

All sizes of routers may be found inside enterprises.[5] The most powerful routers are usually found in ISPs, academic and research facilities. Large businesses may also need more powerful routers to cope with ever increasing demands of intranet data traffic. A three-layer model is in common use, not all of which need be present in smaller networks.[6]

Access

Access routers, including 'small office/home office' (SOHO) models, are located at customer sites such as branch offices that do not need hierarchical routing of their own. Typically, they are optimized for low cost. Some SOHO routers are capable of running alternative free Linux-based firmwares like Tomato, OpenWrt or DD-WRT.[7]

A screenshot of the LuCI web interface used by OpenWrt. This page configures Dynamic DNS.

Distribution

Distribution routers aggregate traffic from multiple access routers, either at the same site, or to collect the data streams from multiple sites to a major enterprise location. Distribution routers are often responsible for enforcing quality of service across a WAN, so they may have considerable memory installed, multiple WAN interface connections, and substantial onboard data processing routines. They may also provide connectivity to groups of file servers or other external networks.

Security

External networks must be carefully considered as part of the overall security strategy. Separate from the router may be a firewall or VPN handling device, or the router may include these and other security functions. Many companies produced security-oriented routers, including Cisco Systems' PIX and ASA5500 series, Juniper's Netscreen, Watchguard's Firebox, Barracuda's variety of mail-oriented devices, and many others.

Core

In enterprises, a core router may provide a "collapsed backbone" interconnecting the distribution tier routers from multiple buildings of a campus, or large enterprise locations. They tend to be optimized for high bandwidth.[8]

Internet connectivity and internal use

Routers intended for ISP and major enterprise connectivity usually exchange routing information using the Border Gateway Protocol (BGP). RFC 4098[9] standard defines the types of BGP-protocol routers according to the routers' functions:

- *Edge router*: Also called a Provider Edge router, is placed at the edge of an ISP network. The router uses External BGP to EBGP protocol routers in other ISPs, or a large enterprise Autonomous System.
- *Subscriber edge router*: Also called a Customer Edge router, is located at the edge of the subscriber's network, it also uses EBGP protocol to its provider's Autonomous System. It is typically used in an (enterprise) organization.
- *Inter-provider border router*: Interconnecting ISPs, is a BGP-protocol router that maintains BGP sessions with other BGP protocol routers in ISP Autonomous Systems.
- Core router: A *core router* resides within an Autonomous System as a back bone to carry traffic between edge routers.[10]
- Within an ISP: In the ISPs Autonomous System, a router uses internal BGP protocol to communicate with other ISP edge routers, other intranet core routers, or the ISPs intranet provider border routers.
- "Internet backbone:" The Internet no longer has a clearly identifiable backbone, unlike its predecessor networks. See default-free zone (DFZ). The major ISPs system routers make up what could be considered to be the current Internet backbone core.[11] ISPs operate all four types of the BGP-protocol routers described here. An ISP "core" router is used to interconnect its edge and border routers. Core routers may also have specialized functions in virtual private networks based on a combination of BGP and Multi-Protocol Label Switching protocols.[12]
- Port forwarding: Routers are also used for port forwarding between private internet connected servers.[5]
- Voice/Data/Fax/Video Processing Routers: Commonly referred to as access servers or gateways, these devices are used to route and process voice, data, video, and fax traffic on the internet. Since 2005, most long-distance phone calls have been processed as IP traffic (VOIP) through a voice gateway. Voice traffic that the traditional cable networks once carried. Use of access server type routers expanded with the advent of the internet, first with dial-up access, and another resurgence with voice phone service.

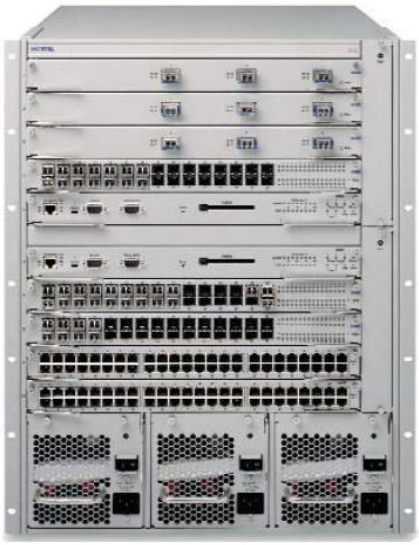
Historical and technical information

The very first device that had fundamentally the same functionality as a router does today, was the Interface Message Processor (IMP); IMPs were the devices that made up the ARPANET, the first packet network. The idea for a router (called "gateways" at the time) initially came about through an international group of computer networking researchers called the International Network Working Group (INWG). Set up in 1972 as an informal group to consider the technical issues involved in connecting different networks, later that year it became a subcommittee of the International Federation for Information Processing.[13]

Leonard Kleinrock and the first IMP.

These devices were different from most previous packet networks in two ways. First, they connected dissimilar kinds of networks, such as serial lines and local area networks. Second, they were connectionless devices, which had no role in assuring that traffic was delivered reliably, leaving that entirely to the hosts (this particular idea had been previously pioneered in the CYCLADES network).

The idea was explored in more detail, with the intention to produce a prototype system, as part of two contemporaneous programs. One was the initial DARPA-initiated program, which created the TCP/IP architecture in use today.[14] The other was a program at Xerox PARC to explore new networking technologies, which produced the PARC Universal Packet system, due to corporate intellectual property concerns it received little attention outside Xerox for years.[15]



Avaya ERS 8600 (2010)

Some time after early 1974 the first Xerox routers became operational. The first true IP router was developed by Virginia Strazisar at BBN, as part of that DARPA-initiated effort, during 1975-1976. By the end of 1976, three PDP-11-based routers were in service in the experimental prototype Internet.[16]

The first multiprotocol routers were independently created by staff researchers at MIT and Stanford in 1981; the Stanford router was done by William Yeager, and the MIT one by Noel Chiappa; both were also based on PDP-11s.[17] [18] [19] [20]

Virtually all networking now uses TCP/IP, but multiprotocol routers are still manufactured. They were important in the early stages of the growth of computer networking, when protocols other than TCP/IP were in use. Modern Internet routers that handle both IPv4 and IPv6 are multiprotocol, but are simpler devices than routers processing AppleTalk, DECnet, IP, and Xerox protocols.

From the mid-1970s and in the 1980s, general-purpose mini-computers served as routers. Modern high-speed routers are highly specialized computers with extra hardware added to speed both common routing functions, such as packet forwarding, and specialised functions such as IPsec encryption.

There is substantial use of Linux and Unix software based machines, running open source routing code, for research and other applications. Cisco's operating system was independently designed. Major router operating systems, such as those from Juniper Networks and Extreme Networks, are extensively modified versions of Unix software.

Forwarding

For pure Internet Protocol (IP) forwarding function, a router is designed to minimize the state information associated with individual packets. The main purpose of a router is to connect multiple networks and forward packets destined either for its own networks or other networks. A router is considered a Layer 3 device because its primary forwarding decision is based on the information in the Layer 3 IP packet, specifically the destination IP address. This process is known as routing. When each router receives a packet, it searches its routing table to find the best match between the destination IP address of the packet and one of the network addresses in the routing table. Once a match is found, the packet is encapsulated in the Layer 2 data link frame for that outgoing interface. A router does not look into the actual data contents that the packet carries, but only at the layer 3 addresses to make a forwarding decision, plus optionally other information in the header for hint on, for example, QoS. Once a packet is forwarded, the router does not retain any historical information about the packet, but the forwarding action can be collected into the statistical data, if so configured.

Forwarding decisions can involve decisions at layers other than layer 3. A function that forwards based on layer 2 information, is properly called a bridge. This function is referred to as layer 2 bridging, as the addresses it uses to forward the traffic are layer 2 addresses (e.g. MAC addresses on Ethernet).

Besides making decision as which interface a packet is forwarded to, which is handled primarily via the routing table, a router also has to manage congestion, when packets arrive at a rate higher than the router can process. Three policies commonly used in the Internet are tail drop, random early detection (RED), and weighted random early detection (WRED). Tail drop is the simplest and most easily implemented; the router simply drops packets once the length of the queue exceeds the size of the buffers in the router. RED probabilistically drops datagrams early when the queue exceeds a pre-configured portion of the buffer, until a pre-determined max, when it becomes tail drop. WRED requires a weight on the average queue size to act upon when the traffic is about to exceed the pre-configured size, so that short bursts will not trigger random drops.

Another function a router performs is to decide which packet should be processed first when multiple queues exist. This is managed through quality of service (QoS), which is critical when Voice over IP is deployed, so that delays between packets do not exceed 150ms to maintain the quality of voice conversations.

Yet another function a router performs is called policy-based routing where special rules are constructed to override the rules derived from the routing table when a packet forwarding decision is made.

These functions may be performed through the same internal paths that the packets travel inside the router. Some of the functions may be performed through an application-specific integrated circuit (ASIC) to avoid overhead caused by multiple CPU cycles, and others may have to be performed through the CPU as these packets need special attention that cannot be handled by an ASIC.

References

[1] "Overview Of Key Routing Protocol Concepts: Architectures, Protocol Types, Algorithms and Metrics" (http://www.tcpipguide.com/free/t_OverviewOfKeyRoutingProtocolConceptsArchitecturesP.htm). Tcpipguide.com. . Retrieved 15 January 2011.

[2] Requirements for IPv4 Routers (ftp://ftp.rfc-editor.org/in-notes/rfc1812.txt),RFC 1812, F. Baker, June 1995

[3] Requirements for Separation of IP Control and Forwarding (ftp://ftp.rfc-editor.org/in-notes/rfc3654.txt),RFC 3654, H. Khosravi & T. Anderson, November 2003

[4] "Setting uo Netflow on Cisco Routers" (http://my-technet.com/index.php/cisco/setting-up-netflow-on-cisco-routers/). MY-Technet.com date unknown. . Retrieved 15 January 2011.

[5] "Windows Home Server: Router Setup" (http://social.technet.microsoft.com/wiki/contents/articles/windows-home-server-router-setup.aspx). Microsoft Technet 14 Aug 2010. . Retrieved 15 January 2011.

[6] Oppenheimer, Pr (2004). *Top-Down Network Design*. Indianapolis: Cisco Press. ISBN 1587051524.

[7] "Windows Small Business Server 2008: Router Setup" (http://social.technet.microsoft.com/wiki/contents/articles/windows-small-business-server-2008-router-setup.aspx). Microsoft Technet Nov 2010. . Retrieved 15 january 2011.

[8] "Core Network Planning" (http://technet.microsoft.com/en-us/library/dd894458(WS.10).aspx). Microsoft Technet May 28, 2009. . Retrieved 15 January 2011.

[9] Terminology for Benchmarking BGP Device Convergence in the Control Plane (ftp://ftp.rfc-editor.org/in-notes/rfc4098.txt),RFC 4098, H. Berkowitz *et al.*,June 2005
[10] "M160 Internet Backbone Router" (http://www.juniper.net/techpubs/qrc/m160-qrc.pdf). Juniper Networks Date unknown. . Retrieved 15 January 2011.
[11] "Virtual Backbone Routers" (http://www.telecomsportal.com/Assets_papers/Routers_&_Netman/Ironbridge_Virt_Bbone_Route.pdf). IronBridge Networks, Inc. September, 2000. . Retrieved 15 January 2011.
[12] BGP/MPLS VPNs (ftp://ftp.rfc-editor.org/in-notes/rfc2547.txt),RFC 2547, E. Rosen and Y. Rekhter, April 2004
[13] Davies, Shanks, Heart, Barker, Despres, Detwiler, and Riml, "Report of Subgroup 1 on Communication System", INWG Note #1.
[14] Vinton Cerf, Robert Kahn, "A Protocol for Packet Network Intercommunication" (http://ieeexplore.ieee.org/iel5/8159/23818/01092259.pdf), IEEE Transactions on Communications, Volume 22, Issue 5, May 1974, pp. 637 - 648.
[15] David Boggs, John Shoch, Edward Taft, Robert Metcalfe, "Pup: An Internetwork Architecture" (http://ieeexplore.ieee.org/iel5/8159/23925/01094684.pdf), IEEE Transactions on Communications, Volume 28, Issue 4, April 1980, pp. 612- 624.
[16] Craig Partridge, S. Blumenthal, "Data networking at BBN" (http://ieeexplore.ieee.org/iel5/85/33687/01603444.pdf); IEEE Annals of the History of Computing, Volume 28, Issue 1; January–March 2006.
[17] Valley of the Nerds: Who Really Invented the Multiprotocol Router, and Why Should We Care? (http://www.pbs.org/cringely/pulpit/1998/pulpit_19981210_000593.html), Public Broadcasting Service, Accessed August 11, 2007.
[18] Router Man (http://www.networkworld.com/supp/2006/anniversary/032706-routerman.html?t5), NetworkWorld, Accessed June 22, 2007.
[19] David D. Clark, "M.I.T. Campus Network Implementation", CCNG-2, Campus Computer Network Group, M.I.T., Cambridge, 1982; pp. 26.
[20] Pete Carey, "A Start-Up's True Tale: Often-told story of Cisco's launch leaves out the drama, intrigue", San Jose Mercury News, December 1, 2001.

External links

- Internet Engineering Task Force, the Routing Area (http://www.ietf.org/html.charters/wg-dir.html#Routing Area) last checked 21 January 2011.
- Internet Corporation for Assigned Names and Numbers (http://www.icann.org/)
- North American Network Operators Group (http://www.nanog.org/)
- Réseaux IP Européens (European IP Networks) (http://www.ripe.net/)
- American Registry for Internet Numbers (http://www.arin.net/)
- Router Default IP and Username Database (http://www.routeripaddress.com/)
- Asia-Pacific Network Information Center (http://www.apnic.net/)
- Latin American Network Information Center (http://www.lacnic.net/)
- African Region Internet Registry (http://www.afrinic.net/)

Self-synchronizing code

In telecommunications, a **self-synchronizing code**[1] is a line code in which the symbol stream formed by a portion of one code word, or by the overlapped portion of any two adjacent code words, is not a valid code word. Put another way, a set of strings (called "code words") over an alphabet is called a self-synchronizing code if for each string obtained by concatenating two code words, the substring starting at the second symbol and ending at the second-last symbol does not contain any code word as substring.

A synonym for self-synchronizing code is **comma-free code**.[2] However, sometimes the term *comma-free code* is used in the meaning of prefix code.[3] The latter is a broader concept: every self-synchronizing code is a prefix code, but not all prefix codes are self-synchronizing.

A self-synchronizing code permits the proper framing of transmitted code words provided that no uncorrected errors occur in the symbol stream; external synchronization is not required. Self-synchronizing codes also allow recovery from uncorrected errors in the stream; with most prefix codes, an uncorrected error in a single bit may propagate errors further in the stream and make the subsequent data unreadable.

Examples

- High-Level Data Link Control (HDLC)
- Advanced Data Communication Control Procedures (ADCCP)
- UTF-8

Notes

[1] US Federal Standard 1037C
[2] Berstel, Jean; Perrin, Dominique (1985), *Theory of Codes*, Academic Press
[3] US Federal Standard 1037C

References

- *This article incorporates public domain material from the General Services Administration* document "Federal Standard 1037C" (http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm) (in support of MIL-STD-188).

Shared medium

In telecommunication, a **shared medium** is a medium or channel of information transfer that serves more than one user at the same time.

Most channels only function correctly when one user is transmitting, so a channel access method is always in effect.

In circuit switching, each user typically gets a fixed share of the channel capacity. A multiplexing scheme divides up the capacity of the medium. Common multiplexing schemes include time-division multiplexing and frequency-division multiplexing. Channel access methods for circuit switching include time division multiple access, frequency-division multiple access, etc.

In packet switching, the sharing is more dynamic — each user takes up little or none of the capacity when idle, and can utilize the entire capacity if transmitting while all other users are idle. Channel access methods for packet switching include carrier sense multiple access, token passing, etc.

Spanning Tree Protocol

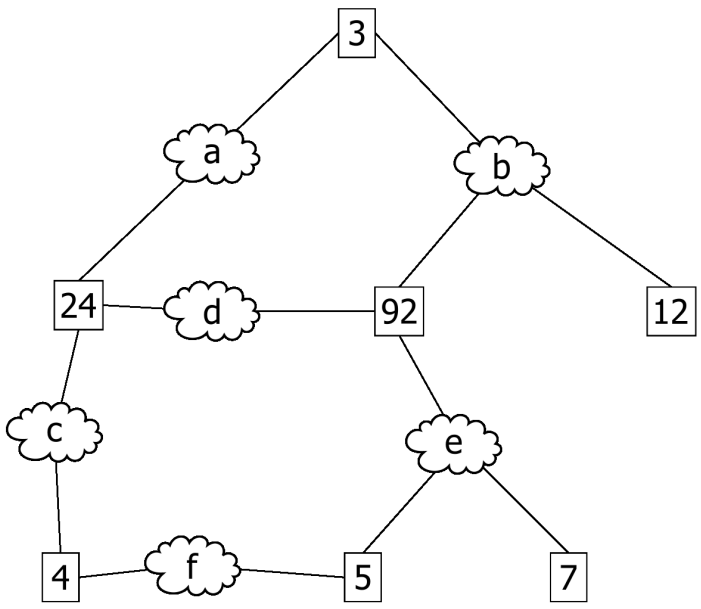
The **Spanning Tree Protocol** (**STP**) is a network protocol that ensures a loop-free topology for any bridged Ethernet local area network. The basic function of STP is to prevent bridge loops and ensuring broadcast radiation. Spanning tree also allows a network design to include spare (redundant) links to provide automatic backup paths if an active link fails, without the danger of bridge loops, or the need for manual enabling/disabling of these backup links.

Spanning Tree Protocol (STP) is standardized as IEEE 802.1D. As the name suggests, it creates a spanning tree within a mesh network of connected layer-2 bridges (typically Ethernet switches), and disables those links that are not part of the spanning tree, leaving a single active path between any two network nodes.

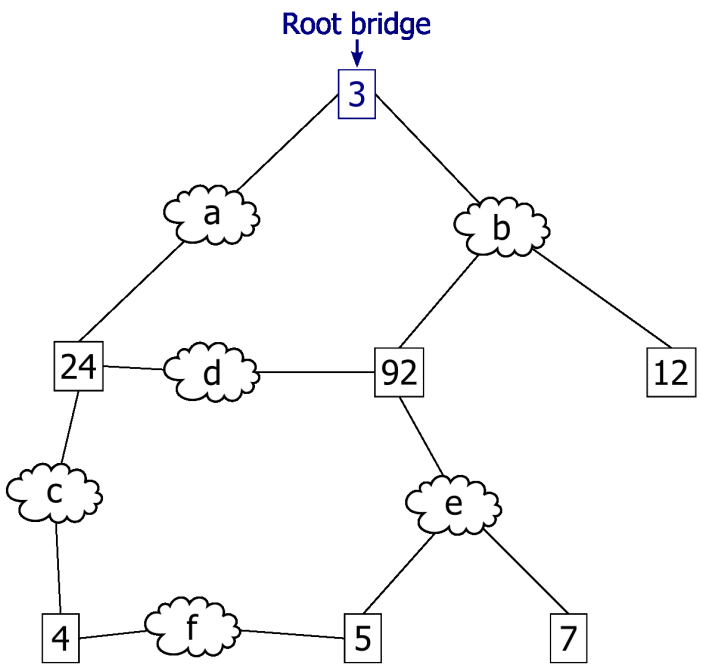
STP is based on an algorithm invented by Radia Perlman while working for Digital Equipment Corporation.[1] [2]

Protocol operation

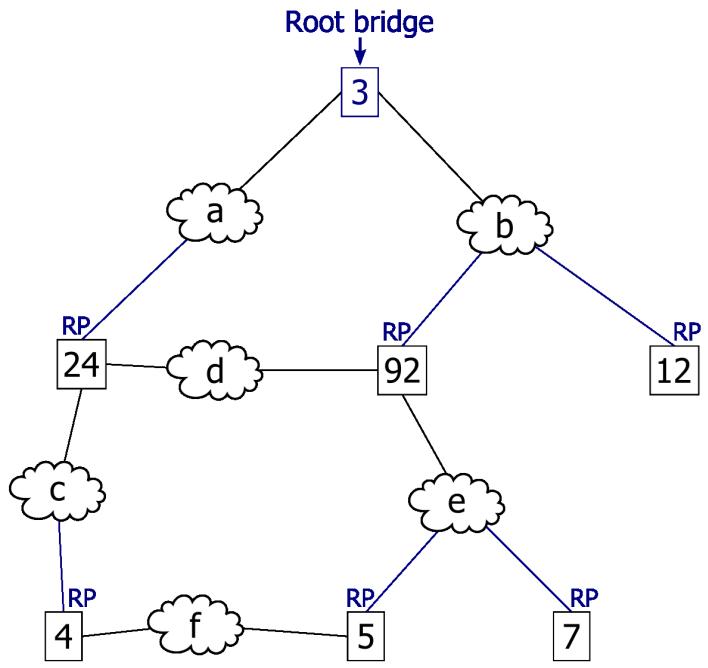
The collection of bridges in a local area network (LAN) can be depicted as a graph whose nodes are bridges and LAN segments (or cables), and whose edges are the interfaces connecting the bridges to the segments. To break loops in the LAN while maintaining access to all LAN segments, the bridges collectively compute a spanning tree. The spanning tree is not necessarily a minimum cost spanning tree. A network administrator can reduce the cost of a spanning tree, if necessary, by altering some of the configuration parameters in such a way as to affect the choice of the root of the spanning tree. The spanning tree that the bridges compute using the Spanning Tree Protocol can be determined using the following rules. The example network at the right, below, will be used to illustrate the rules.



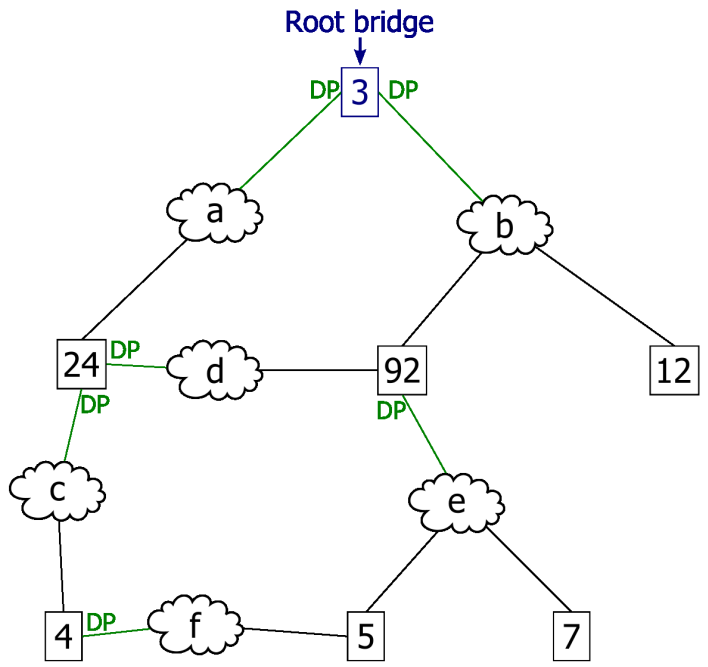
1. An example network. The numbered boxes represent bridges (the number represents the bridge ID). The lettered clouds represent network segments.



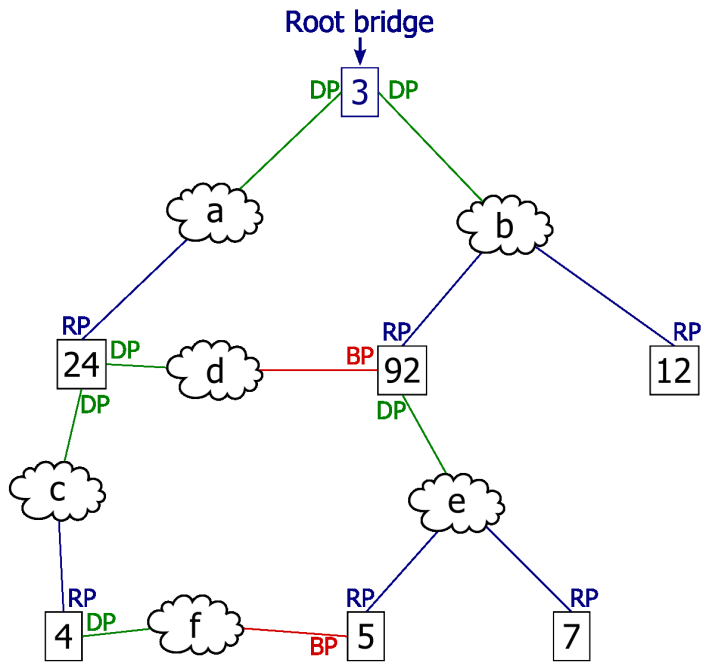
2. The smallest bridge ID is 3. Therefore, bridge 3 is the root bridge.



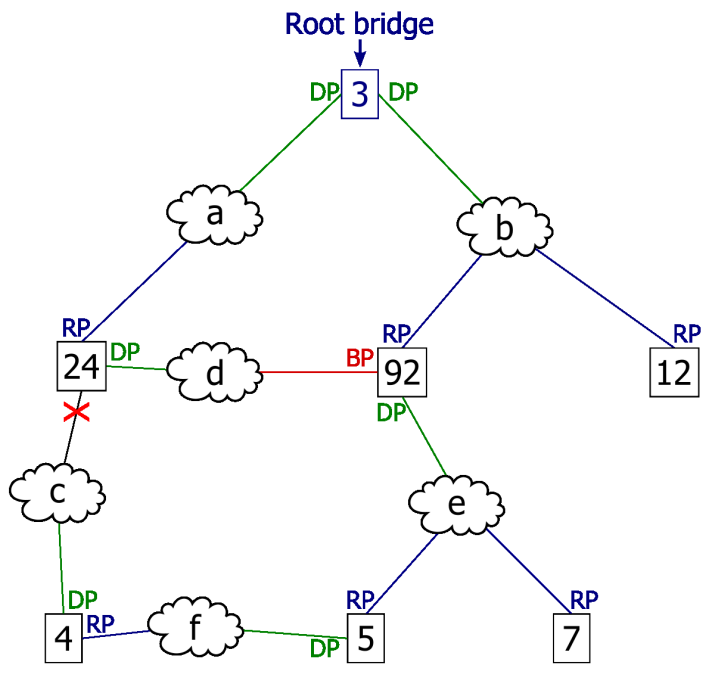
3. Assuming that the cost of traversing any network segment is 1, the least cost path from bridge 4 to the root bridge goes through network segment c. Therefore, the root port for bridge 4 is the one on network segment c.



4. The least cost path to the root from network segment e goes through bridge 92. Therefore the designated port for network segment e is the port that connects bridge 92 to network segment e.



5. This diagram illustrates all port states as computed by the spanning tree algorithm. Any active port that is not a root port or a designated port is a blocked port.



6. After link failure the spanning tree algorithm computes and spans new least-cost tree.

Select a root bridge. The *root bridge* of the spanning tree is the bridge with the smallest (lowest) bridge ID. Each bridge has a unique identifier (ID) and a configurable priority number; the bridge ID contains both numbers. To compare two bridge IDs, the priority is compared first. If two bridges have equal priority, then the MAC addresses

are compared. For example, if switches A (MAC=0200.0000.1111) and B (MAC=0200.0000.2222) both have a priority of 10, then switch A will be selected as the root bridge. If the network administrators would like switch B to become the root bridge, they must set its priority to be less than 10.

Determine the least cost paths to the root bridge. The computed spanning tree has the property that messages from any connected device to the root bridge traverse a least cost path, i.e., a path from the device to the root that has minimum cost among all paths from the device to the root. The cost of traversing a path is the sum of the costs of the segments on the path. Different technologies have different default costs for network segments. An administrator can configure the cost of traversing a particular network segment. The property that messages always traverse least-cost paths to the root is guaranteed by the following two rules.

Least cost path from each bridge. After the root bridge has been chosen, each bridge determines the cost of each possible path from itself to the root. From these, it picks one with the smallest cost (a least-cost path). The port connecting to that path becomes the *root port* (RP) of the bridge.

Least cost path from each network segment. The bridges on a network segment collectively determine which bridge has the least-cost path from the network segment to the root. The port connecting this bridge to the network segment is then the *designated port* (DP) for the segment.

Disable all other root paths. Any active port that is not a root port or a designated port is a *blocked port* (BP).

Modifications in case of ties. The above rules over-simplify the situation slightly, because it is possible that there are ties, for example, two or more ports on a single bridge are attached to least-cost paths to the root or two or more bridges on the same network segment have equal least-cost paths to the root. To break such ties:

Breaking ties for root ports. When multiple paths from a bridge are least-cost paths, the chosen path uses the neighbor bridge with the lower bridge ID. The root port is thus the one connecting to the bridge with the lowest bridge ID. For example, in figure 3, if switch 4 were connected to network segment d, there would be two paths of length 2 to the root, one path going through bridge 24 and the other through bridge 92. Because there are two least cost paths, the lower bridge ID (24) would be used as the tie-breaker in choosing which path to use.

Breaking ties for designated ports. When more than one bridge on a segment leads to a least-cost path to the root, the bridge with the lower bridge ID is used to forward messages to the root. The port attaching that bridge to the network segment is the *designated port* for the segment. In figure 4, there are two least cost paths from network segment d to the root, one going through bridge 24 and the other through bridge 92. The lower bridge ID is 24, so the tie breaker dictates that the designated port is the port through which network segment d is connected to bridge 24. If bridge IDs were equal, then the bridge with the lowest MAC address would have the designated port. In either case, the loser sets the port as being blocked.

The final tie-breaker. In some cases, there may still be a tie, as when two bridges are connected by multiple cables. In this case, multiple ports on a single bridge are candidates for root port. In this case, the path which passes through the port on the neighbor bridge that has the lowest port priority is used.

In summary, the sequence of events to determine the best received BPDU (which is your best path to the root) is

1. lowest root bridge id
2. lowest root path cost
3. lowest sender bridge id
4. lowest sender port number

Data rate and STP path cost

Data rate	STP Cost (802.1D-1998)	RSTP Cost (802.1W-2001)
4 Mbit/s	250	5,000,000
10 Mbit/s	100	2,000,000
16 Mbit/s	62	1,250,000
100 Mbit/s	19	200,000
1 Gbit/s	4	20,000
2 Gbit/s	3	10,000
10 Gbit/s	2	2,000

Bridge Protocol Data Units (BPDUs)

The above rules describe one way of determining what spanning tree will be computed by the algorithm, but the rules as written require knowledge of the entire network. The bridges have to determine the root bridge and compute the port roles (root, designated, or blocked) with only the information that they have. To ensure that each bridge has enough information, the bridges use special data frames called **Bridge Protocol Data Units** (BPDUs) to exchange information about bridge IDs and root path costs.

A bridge sends a BPDU frame using the unique MAC address of the port itself as a source address, and a destination address of the STP multicast address 01:80:C2:00:00:00.

There are three types of BPDUs:

- Configuration BPDU (CBPDU), used for Spanning Tree computation
- Topology Change Notification (TCN) BPDU, used to announce changes in the network topology
- Topology Change Notification Acknowledgment (TCA)

BPDUs are exchanged regularly (every 2 seconds by default) and enable switches to keep track of network changes and to start and stop forwarding at ports as required.

When a device is first attached to a switch port, it will not immediately start to forward data. It will instead go through a number of states while it processes BPDUs and determines the topology of the network. When a host is attached such as a computer, printer or server the port will always go into the forwarding state, albeit after a delay of about 30 seconds while it goes through the listening and learning states (see below). The time spent in the listening and learning states is determined by a value known as the forward delay (default 15 seconds and set by the root bridge). However, if instead another *switch* is connected, the port may remain in blocking mode if it is determined that it would cause a loop in the network. Topology Change Notification (TCN) BPDUs are used to inform other switches of port changes. TCNs are injected into the network by a non-root switch and propagated to the root. Upon receipt of the TCN, the root switch will set a Topology Change flag in its normal BPDUs. This flag is propagated to all other switches to instruct them to rapidly age out their forwarding table entries.

STP switch port states:

- **Blocking** - A port that would cause a switching loop, no user data is sent or received but it may go into forwarding mode if the other links in use were to fail and the spanning tree algorithm determines the port may transition to the forwarding state. BPDU data is still received in blocking state.
- **Listening** - The switch processes BPDUs and awaits possible new information that would cause it to return to the blocking state.
- **Learning** - While the port does not yet forward frames (packets) it does learn source addresses from frames received and adds them to the filtering database (switching database)

- **Forwarding** - A port receiving and sending data, normal operation. STP still monitors incoming BPDUs that would indicate it should return to the blocking state to prevent a loop.
- **Disabled** - Not strictly part of STP, a network administrator can manually disable a port

To prevent the delay when connecting hosts to a switch and during some topology changes, Rapid STP was developed and standardized by IEEE 802.1w, which allows a switch port to rapidly transition into the forwarding state during these situations.

BPDU fields

The bridge ID, or BID, is a field inside a BPDU packet. It is eight bytes in length. The first two bytes are the Bridge Priority, an unsigned integer of 0-65,535. The last six bytes are a MAC address supplied by the switch. In the event that MAC Address Reduction is used, the first two bytes are used differently. The first four bits are a configurable priority, and the last twelve bits carry either the VLAN ID or MSTP instance number.

Evolutions and extensions

The first spanning tree protocol was invented in 1985 at the Digital Equipment Corporation by Radia Perlman.[1] In 1990, the IEEE published the first standard for the protocol as 802.1D,[3] based on the algorithm designed by Perlman. Subsequent versions were published in 1998[4] and 2004,[5] incorporating various extensions.

Although the purpose of a standard is to promote interworking of equipment from different vendors, different implementations of a standard are not guaranteed to work, due for example to differences in default timer settings. The IEEE encourages vendors to provide a "Protocol Implementation Conformance Statement", declaring which capabilities and options have been implemented,[5] to help users determine whether different implementations will interwork correctly.

Also, the original Perlman-inspired Spanning Tree Protocol, called DEC STP, is not a standard and differs from the IEEE version in message format as well as timer settings. Some bridges implement both the IEEE and the DEC versions of the Spanning Tree Protocol, but their interworking can create issues for the network administrator, as illustrated by the problem discussed in an on-line Cisco document.[6]

Rapid Spanning Tree Protocol (RSTP)

In 2001, the IEEE introduced Rapid Spanning Tree Protocol (RSTP) as 802.1w. RSTP provides significantly faster spanning tree convergence after a topology change, introducing new convergence behaviors and bridge port roles to do this. RSTP was designed to be backwards-compatible with standard STP.

While STP can take 30 to 50 seconds to respond to a topology change, RSTP is typically able to respond to changes within $3 \times$ Hello times (default: 3 times 2 seconds) or within a few milliseconds of a physical link failure. The so-called Hello time is an important and configurable time interval that is used by RSTP for several purposes; its default value is 2 seconds.[7] [8] Standard IEEE 802.1D-2004 now incorporates RSTP and obsoletes the original STP standard.

RSTP Operation

RSTP adds new bridge port roles in order to speed convergence following a link failure.

RSTP bridge port roles:

- **Root** - A forwarding port that is the best port from Nonroot-bridge to Rootbridge
- **Designated** - A forwarding port for every LAN segment
- **Alternate** - An alternate path to the root bridge. This path is different than using the root port.
- **Backup** - A backup/redundant path to a segment where another bridge port already connects.
- **Disabled** - Not strictly part of STP, a network administrator can manually disable a port

Additional RSTP Operation Details:

- Detection of root switch failure is done in 3 hello times, which is 6 seconds if default hello times have not been changed.
- Ports may be configured as edge ports if they are attached to a LAN that has no other bridges attached. These edge ports transition directly to the forwarding state. RSTP still continues to monitor the port for BPDUs in case a bridge is connected. RSTP can also be configured to automatically detect edge ports. As soon as the bridge detects a BPDU coming to an edge port, the port becomes a non-edge port.
- Unlike in STP, RSTP will respond to BPDUs sent from the direction of the root bridge. An RSTP bridge will "propose" its spanning tree information to its designated ports. If another RSTP bridge receives this information and determines this is the superior root information, it sets all its other ports to discarding. The bridge may send an "agreement" to the first bridge confirming its superior spanning tree information. The first bridge, upon receiving this agreement, knows it can rapidly transition that port to the forwarding state bypassing the traditional listening/learning state transition. This essentially creates a cascading effect away from the root bridge where each designated bridge proposes to its neighbors to determine if it can make a rapid transition. This is one of the major elements that allows RSTP to achieve faster convergence times than STP.
- As discussed in the port role details above, RSTP maintains backup details regarding the discarding status of ports. This avoids timeouts if the current forwarding ports were to fail or BPDUs were not received on the root port in a certain interval.
- RSTP will revert to legacy STP on an interface if a legacy version of an STP BPDU is detected on that port.

Per-VLAN Spanning Tree (PVST)

In Ethernet switched environments where multiple Virtual LANs exist, spanning tree can be deployed per Virtual LAN. Cisco's name for this is *per VLAN spanning tree* (PVST and PVST+, which is the default protocol used by Cisco switches). Both PVST and PVST+ protocols are Cisco proprietary protocols and they cannot be used on most 3rd party switches. Some equipment from Force10 Networks, Extreme Networks, and Blade Network Technologies support PVST+.[9] [10] [11] Extreme Networks does so with two limitations (lack of support on ports where the VLAN is untagged/native and also on the VLAN with ID 1). PVST works only with ISL (Cisco's proprietary protocol for VLAN encapsulation) due to its embedded Spanning tree ID. Due to high penetration of the IEEE 802.1Q VLAN trunking standard and PVST's dependence on ISL, Cisco defined a different PVST+ standard for 802.1Q encapsulation. PVST+ can tunnel across an MSTP Region.[12]

VLAN Spanning Tree Protocol (VSTP)

In Juniper Networks environment, if compatibility to Cisco's proprietary PVST protocol is required, VLAN Spanning Tree Protocol (VSTP) can be configured. VSTP maintains a separate spanning-tree instance for each VLAN configured in the switch. The VSTP protocol is only supported by the EX Series from Juniper Networks. There are two restrictions to the compatibility of VSTP:

1. VSTP supports only 253 different spanning-tree topologies. If there are more than 253 VLANs, it is recommended to configure RSTP in addition to VSTP, and VLANs beyond 253 will be handled by the RSTP protocol.
2. MVRP does not support VSTP. If this protocol is in use, VLAN membership for trunk interfaces must be statically configured[13].

By default, VSTP uses the RSTP protocol as its core spanning-tree protocol, but usage of STP can be forced if the network includes old bridges[14].

For more information about configuring the VSTP protocol on Juniper Networks switches, go to the official documentation. Understanding VSTP [14]

Multiple Spanning Tree Protocol (MSTP)

The *Multiple Spanning Tree Protocol* (MSTP), originally defined in IEEE 802.1s and later merged into IEEE 802.1Q-2005, defines an extension to RSTP to further develop the usefulness of virtual LANs (VLANs). This "Per-VLAN" Multiple Spanning Tree Protocol configures a separate Spanning Tree for each VLAN group and blocks all but one of the possible alternate paths within each Spanning Tree.

If there is only one Virtual LAN (VLAN) in the network, single (traditional) STP works appropriately. If the network contains more than one VLAN, the logical network configured by single STP would work, but it is possible to make better use of the alternate paths available by using an alternate spanning tree for different VLANs or groups of VLANs.

MSTP allows formation of MST regions that can run multiple MST instances (MSTI). Multiple regions and other STP bridges are interconnected using one single common spanning tree (CST).

MSTP is similar to Cisco Systems' **Multiple Instances Spanning Tree Protocol** (MISTP), and is an evolution of the Spanning Tree Protocol and the Rapid Spanning Tree Protocol. It was introduced in IEEE 802.1s as an amendment to 802.1Q, 1998 edition. Standard IEEE 802.1Q-2005 now includes MSTP.

Unlike some proprietary per-VLAN spanning tree implementations,[15] MSTP includes all of its spanning tree information in a single BPDU format. Not only does this reduce the number of BPDUs required on a LAN to communicate spanning tree information for each VLAN, but it also ensures backward compatibility with RSTP (and in effect, classic STP too). MSTP does this by encoding additional region information after the standard RSTP BPDU as well as a number of MSTI messages (from 0 to 64 instances, although in practice many bridges support fewer). Each of these MSTI configuration messages conveys the spanning tree information for each instance. Each instance can be assigned a number of configured VLANs and frames (packets) assigned to these VLANs operate in this spanning tree instance whenever they are inside the MST region. In order to avoid conveying their entire VLAN to spanning tree mapping in each BPDU, bridges encode an MD5 digest of their VLAN to instance table in the MSTP BPDU. This digest is then used by other MSTP bridges, along with other administratively configured values, to determine if the neighboring bridge is in the same MST region as itself.

MSTP is fully compatible with RSTP bridges, in that an MSTP BPDU can be interpreted by an RSTP bridge as an RSTP BPDU. This not only allows compatibility with RSTP bridges without configuration changes, but also causes any RSTP bridges outside of an MSTP region to see the region as a single RSTP bridge, regardless of the number of MSTP bridges inside the region itself. In order to further facilitate this view of an MST region as a single RSTP bridge, the MSTP protocol uses a variable known as remaining hops as a time to live counter instead of the message

age timer used by RSTP. The message age time is only incremented once when spanning tree information enters an MST region, and therefore RSTP bridges will see a region as only one "hop" in the spanning tree. Ports at the edge of an MST region connected to either an RSTP or STP bridge or an endpoint are known as boundary ports. As in RSTP, these ports can be configured as edge ports to facilitate rapid changes to the forwarding state when connected to endpoints.

Rapid Per-VLAN Spanning Tree (R-PVST)

Cisco's proprietary protocol that combines the functionalities of RSTP and PVST. It is based on a per-VLAN instance that creates a spanning tree for each VLAN.

References

[1] Perlman, Radia (1985). "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN". *ACM SIGCOMM Computer Communication Review* **15** (4): 44–53. doi:10.1145/318951.319004.

[2] Perlman, Radia (2000). *Interconnections, Second Edition*. USA: Addison-Wesley. ISBN 0-201-63448-1.

[3] LAN/MAN Standards Committee of the IEEE Computer Society, ed (1990). *ANSI/IEEE Std 802.1D*. IEEE

[4] LAN/MAN Standards Committee of the IEEE Computer Society, ed (1998). *ANSI/IEEE Std 802.1D, 1998 Edition, Part 3: Media Access Control (MAC) Bridges*. IEEE

[5] LAN/MAN Standards Committee of the IEEE Computer Society, ed (2004). *ANSI/IEEE Std 802.1D - 2004: IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*. IEEE

[6] (PDF) *Understanding Issues Related to Inter-VLAN Bridging* (http://www.cisco.com/warp/public/473/inter-vlan_11072.pdf). Cisco Systems, Inc.. 11072.

[7] Waldemar Wojdak (March 2003 [CPCI203]). "Rapid Spanning Tree Protocol: A new solution from an old technology" (http://www.compactpci-systems.com/articles/id/?203). . Retrieved 2008-08-04.

[8] "Understanding Rapid Spanning Tree Protocol (802.1w)" (http://www.cisco.com/en/US/tech/tk389/tk621/technologies_white_paper09186a0080094cfa.shtml). . Retrieved 2008-11-27.

[9] "Technical Documentation" (https://www.force10networks.com/CSPortal20/TechTips/0050B_HowDoIConfigureSpanningTree.aspx). Force10Networks. . Retrieved 2011-01-25.

[10] "ExtremeXOS Operating System, Version 12.5" (http://www.extremenetworks.com/libraries/products/DSExtXOS_1030.pdf) (PDF). Extreme Networks. 2010. . Retrieved 2011-01-25.

[11] "BLADE PVST+ Interoperability with Cisco" (http://www.bladenetwork.net/userfiles/file/PDFs/WP_PVST_SpanningTree_Cisco.pdf) (PDF). 2006. . Retrieved 2011-01-25.

[12] "Bridging Between IEEE 802.1Q VLANs" (http://www.cisco.com/en/US/docs/ios/12_1t/12_1t3/feature/guide/dtbridge.html#wp1020686). Cisco Systems. . Retrieved 2011-01-25.

[13] http://www.juniper.net/techpubs/en_US/junos10.0/topics/concept/bridging-mvrp-ex-series.html

[14] https://www.juniper.net/techpubs/en_US/junos9.4/topics/concept/spanning-trees-ex-series-vstp-understanding.html

[15] "CiscoWorks LAN Management Solution 3.2 Deployment Guide" (https://www.cisco.com/en/US/prod/collateral/netmgtsw/ps6504/ps6528/ps2425/white_paper_c07-552114.html#wp9003215). August 2009. . Retrieved 2010-01-25.

External links

- Cisco home page for the Spanning-Tree protocol family (http://www.cisco.com/en/US/tech/tk389/tk621/tsd_technology_support_protocol_home.html) (discusses CST, MISTP, PVST, PVST+, RSTP, STP)
- STP article in the Wireshark wiki (http://wiki.wireshark.org/STP) Includes a sample PCAP-file of captured STP traffic.
- Perlman, Radia. "Algorhyme" (http://web.archive.org/web/20110719212324/http://www.csua.berkeley.edu/~ranga/humor/algorhyme.txt). University of California at Berkeley. Archived from the original (http://www.csua.berkeley.edu/~ranga/humor/algorhyme.txt) on 2011-07-19. Retrieved 2011-09-01.
- IEEE Standards
  - ANSI/IEEE 802.1D-2004 standard (http://standards.ieee.org/getieee802/download/802.1D-2004.pdf), section 17 discusses RSTP (Regular STP is no longer a part of this standard. This is pointed out in section 8.)
  - ANSI/IEEE 802.1Q-2005 standard (http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf), section 13 discusses MSTP

- RFCs
  - RFC 2674-1999, proposed standard, Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions
  - RFC 1525-1993, - SBRIDGEMIB, proposed standard, Definitions of Managed Objects for Source Routing Bridges
  - RFC 1493-1993 - BRIDGEMIB, draft standard, Definitions of Managed Objects for Bridges
- Spanning Tree Direct vs Indirect Link Failures - CCIE Study (http://blog.ipexpert.com/2010/03/22/spanning-tree-direct-vs-indirect-link-failures/)

Tunneling protocol

Computer networks use a **tunneling protocol** when one network protocol (the **delivery protocol**) encapsulates a different **payload protocol**. By using tunneling one can (for example) carry a payload over an incompatible delivery-network, or provide a secure path through an untrusted network.

Tunneling typically contrasts with a layered protocol model such as those of OSI or TCP/IP. The delivery protocol usually (but not always) operates at a higher level in the model than does the payload protocol, or at the same level.

To understand a particular protocol stack, network engineers must understand both the payload and delivery protocol sets.

As an example of network layer over network layer, Generic Routing Encapsulation (GRE), a protocol running over IP (IP Protocol Number 47), often serves to carry IP packets, with RFC 1918 private addresses, over the Internet using delivery packets with public IP addresses. In this case, the delivery and payload protocols are compatible, but the payload addresses are incompatible with those of the delivery network.

In contrast, an IP payload might believe it sees a data link layer delivery when it is carried inside the Layer 2 Tunneling Protocol (L2TP), which appears to the payload mechanism as a protocol of the data link layer. L2TP, however, actually runs over the transport layer using User Datagram Protocol (UDP) over IP. The IP in the delivery protocol could run over any data-link protocol from IEEE 802.2 over IEEE 802.3 (i.e., standards-based Ethernet) to the Point-to-Point Protocol (PPP) over a dialup modem link.

Tunneling protocols may use data encryption to transport insecure payload protocols over a public network (such as the Internet), thereby providing VPN functionality. IPsec has an end-to-end Transport Mode, but can also operate in a tunneling mode through a trusted security gateway.

Secure shell tunneling

A secure shell (SSH) tunnel consists of an encrypted tunnel created through a SSH protocol connection. Users may set up SSH tunnels to transfer unencrypted traffic over a network through an encrypted channel. For example, Microsoft Windows machines can share files using the Server Message Block (SMB) protocol, a non-encrypted protocol. If one were to mount a Microsoft Windows file-system remotely through the Internet, someone snooping on the connection could see transferred files. To mount the Windows file-system securely, one can establish an SSH tunnel that routes all SMB traffic to the remote fileserver through an encrypted channel. Even though the SMB protocol itself contains no encryption, the encrypted SSH channel through which it travels offers security.

To set up an SSH tunnel, one configures an SSH client to forward a specified local port to a port on the remote machine. Once the SSH tunnel has been established, the user can connect to the specified local port to access the network service. The local port need not have the same port number as the remote port.

SSH tunnels provide a means to bypass firewalls that prohibit certain Internet services — so long as a site allows outgoing connections. For example, an organization may prohibit a user from accessing Internet web pages (port 80)

directly without passing through the organization's proxy filter (which provides the organization with a means of monitoring and controlling what the user sees through the web). But users may not wish to have their web traffic monitored or blocked by the organization's proxy filter. If users can connect to an external SSH server, they can create an SSH tunnel to forward a given port on their local machine to port 80 on a remote web-server. To access the remote web-server, users would point their browser to the local port at http://localhost/.

Some SSH clients support dynamic port forwarding that allows the user to create a SOCKS 4/5 proxy. In this case users can configure their applications to use their local SOCKS proxy server. This gives more flexibility than creating an SSH tunnel to a single port as previously described. SOCKS can free the user from the limitations of connecting only to a predefined remote port and server. If an application doesn't support SOCKS, one can use a "socksifier" to redirect the application to the local SOCKS proxy server. Some "socksifiers" support SSH directly, thus avoiding the need for an SSH client.

Tunneling to circumvent firewall policy

Users can also use tunneling to "sneak through" a firewall, using a protocol that the firewall would normally block, but "wrapped" inside a protocol that the firewall does not block, such as HTTP. If the firewall policy does not specifically exclude this kind of "wrapping", this trick can function to get around the intended firewall policy.

Another HTTP-based tunneling method uses the HTTP CONNECT method/command. A client issues the HTTP CONNECT command to a HTTP proxy. The proxy then makes a TCP connection to a particular server:port, and relays data between that server:port and the client connection. Because this creates a security hole, CONNECT-capable HTTP proxies commonly restrict access to the CONNECT method. The proxy allows access only to a whitelist of specific authorized servers.

External links

- HOWTO: Set up a Windows SSH server for VNC tunneling [1]
- SSH VPN tunnel, see the SSH-BASED VIRTUAL PRIVATE NETWORKS section [2]

This article was originally based on material from the Free On-line Dictionary of Computing, which is licensed under the GFDL.

References

[1] http://www.allthingsmarked.com/2006/08/17/how-to-set-up-a-windows-ssh-server-for-vnc-tunneling/
[2] http://www.freebsd.org/cgi/man.cgi?query=ssh

Type of service

The **type of service** (**TOS**) field in the IPv4 header has had various purposes over the years, and has been defined in different ways by five RFCs[1] . The modern redefinition of the TOS field is a six-bit differentiated services code point (DSCP) field and a two-bit Explicit Congestion Notification field.

Type of service

The original intention of the TOS field was for a sending host to specify a preference for how the datagram would be handled as it made its way through an internet. For instance, one host could set its IPv4 datagrams' TOS field value to prefer low delay, while another might prefer high reliability. In practice, the TOS field was not widely implemented. However, a great deal of experimental, research and deployment work has focused on how to make use of these eight bits, resulting in the current DSCP field definition.

As defined in RFC 791, the following eight bits were allocated to a TOS field in the IP header. They can be divided into 5 subfields:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Precedence			D	T	R	Unused	

0 = Most significant Bit; 7 = Least significant Bit.

The three precedence bits have a from 0 to 7 and are used to indicate the importance of a datagram (default is 000, higher is better).

Precedence	Hexadecimal	Decimal	Decimal of 6 left bits	Description
111	0xE0	224	56	Network control
110	0xC0	192	48	Internetwork control
101	0xA0	160	40	CRITIC/ECP
100	0x80	128	32	Flash override
011	0x60	96	24	Flash
010	0x40	64	16	Immediate
001	0x20	32	8	Priority
000	0x00	0	0	Routine

The decimal and hexadecimal values above, assumes bits 3 to 7 are set to 0.

Bits 3 4 5 represent the following:

- D: requests low delay
- T: requests high throughput
- R: requests high reliability

Bit 6 was later defined by RFC 1349 as "Minimize monetary cost".

A router maintains a TOS value for each route in its routing table. Routes learned through a protocol that does not support TOS are assigned a TOS of zero. Routers use the TOS to choose a destination for the packet.

1. The router locates in its routing table all available routes to the destination.
2. If one or more of those routes have a TOS that exactly matches the TOS specified in the packet, the router chooses the route with the best metric.
3. Otherwise, the router repeats the above step, except looking at routes whose TOS is zero.

4. If no route was chosen above, the router drops the packet because the destination is unreachable. The router returns an ICMP Destination Unreachable error specifying the appropriate code: either Network Unreachable with Type of Service (code 11) or Host Unreachable with Type of Service (code 12).

The TOS model of the "IP precedence" model as defined in RFC 791 and RFC 1349 are outdated. The replacing model as described in RFC 2474 uses the DSCP field in the IPv4 header and IPv6 Header.[2] Some backwards compatibility is maintained.

References

[1] RFC 791, RFC 1122, RFC 1349, RFC 2474, and RFC 3168. For a full history of the TOS field, see section 22 of RFC 3168.
[2] asterisk's docs – doc/ip-tos.txt (http://svn.digium.com/view/asterisk/branches/1.4/doc/ip-tos.txt?revision=58955&view=markup)

Further reading

- John Evans, Clarence Filsfils (2007). *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice*. Morgan Kaufmann. ISBN 0-370561-370548-5.

External links

- Linux Advanced Routing & Traffic Control. How to set TOS byte via IPChains (http://lartc.org/howto/index.html)

Virtual circuit

In telecommunications and computer networks, a **virtual circuit** (**VC**), synonymous with **virtual connection** and virtual channel, is a connection oriented communication service that is delivered by means of packet mode communication. After a connection or virtual circuit is established between two nodes or application processes, a bit stream or byte stream may be delivered between the nodes; a virtual circuit protocol allows higher level protocols to avoid dealing with the division of data into segments, packets, or frames.

Virtual circuit communication resembles circuit switching, since both are connection oriented, meaning that in both cases data is delivered in correct order, and signalling overhead is required during a connection establishment phase. However, circuit switching provides constant bit rate and latency, while these may vary in a virtual circuit service because of reasons such as:

- varying packet queue lengths in the network nodes,
- varying bit rate generated by the application,
- varying load from other users sharing the same network resources by means of statistical multiplexing, etc.

Many virtual circuit protocols, but not all, provide reliable communication service, by means of data retransmissions because of error detection and automatic repeat request (ARQ).

Layer 4 virtual circuits

Connection oriented transport layer datalink protocols such as TCP[1] [2] may rely on a connectionless packet switching network layer protocol such as IP, where different packets may be routed over different paths, and thus be delivered out of order. However, a virtual circuit[2] [3] [4] is possible since TCP includes segment numbering and reordering on the receiver side to prevent out-of-order delivery.

Layer 2/3 virtual circuits

Network layer and datalink layer virtual circuit protocols are based on connection oriented packet switching, meaning that data is always delivered along the same network path, i.e. through the same nodes. Advantages with this over connectionless packet switching are:

- Bandwidth reservation during the connection establishment phase is supported, making guaranteed Quality of Service (QoS) possible. For example, a constant bit rate QoS class may be provided, resulting in emulation of circuit switching.
- Less overhead is required, since the packets are not routed individually and complete addressing information is not provided in the header of each data packet. Only a small virtual channel identifier (VCI) is required in each packet. Routing information is only transferred to the network nodes during the connection establishment phase.
- The network nodes are faster and have higher capacity in theory, since they are switches that only perform routing during the connection establishment phase, while connectionless network nodes are routers that perform routing for each packet individually. Switching only involves looking up the virtual channel identifier in a table rather than analyzing a complete address. Switches can easily be implemented in ASIC hardware, while routing is more complex and requires software implementation. However, because of the large market of IP routers, and because advanced IP routers support layer 3 switching, modern IP routers may today be faster than switches for connection oriented protocols.

Examples of protocols that provide virtual circuits

Examples of transport layer protocols that provide a virtual circuit:

- Transmission Control Protocol (TCP), where a reliable virtual circuit is established on top of the underlying unreliable and connectionless IP protocol. The virtual circuit is identified by the source and destination network socket address pair, i.e. the sender and receiver IP address and port number. Guaranteed QoS is not provided.
- Stream Control Transmission Protocol (SCTP), where a virtual circuit is established on top of either the IP protocol or the UDP protocol.

Examples of network layer and datalink layer virtual circuit protocols, where data always is delivered over the same path:

- X.25, where the VC is identified by a virtual channel identifier (VCI). X.25 provides reliable node-to-node communication and guaranteed QoS.
- Frame relay, where the VC is identified by a VCI. Frame relay is unreliable, but may provide guaranteed QoS.
- Asynchronous Transfer Mode (ATM), where the circuit is identified by a virtual path identifier (VPI) and virtual channel identifier (VCI) pair. ATM is unreliable, but may provide guaranteed QoS.
- General Packet Radio Service (GPRS)
- Multiprotocol label switching (MPLS), which can be used for IP over virtual circuits. Each circuit is identified by a label. MPLS is unreliable, but provides eight different QoS classes.

Permanent and switched virtual circuits in ATM, frame relay, and X.25

Switched virtual circuits (**SVCs**) are generally set up on a per-call basis and are disconnected when the call is terminated; however, a **permanent virtual circuit** (**PVC**) can be established as an option to provide a dedicated circuit link between two facilities. PVC configuration is usually preconfigured by the service provider. Unlike SVCs, PVC are usually very seldom broken/disconnected.

A switched virtual circuit (SVC) is a virtual circuit that is dynamically established on demand and is torn down when transmission is complete, for example after a phone call or a file download. SVCs are used in situations where data transmission is sporadic and/or not always between the same data terminal equipment (DTE) endpoints.

A permanent virtual circuit (PVC) is a virtual circuit established for repeated/continuous use between the same DTE. In a PVC, the long-term association is identical to the data transfer phase of a virtual call. Permanent virtual circuits eliminate the need for repeated call set-up and clearing.

- Frame relay is typically used to provide PVCs.
- ATM provides both **switched virtual connections** and **permanent virtual connections**, as they are called in ATM terminology.
- X.25 provides both **virtual calls** and PVCs, although not all X.25 service providers or DTE implementations support PVCs as their use was much less common than SVCs

References

[1] RFC 793
[2] RFC 1180
[3] RFC 955
[4] RFC 1644

Internet Layer

Autonomous system (Internet)

Within the Internet, an **Autonomous System** (**AS**) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet.[1]

Originally the definition required control by a single entity, typically an Internet service provider or a very large organization with independent connections to multiple networks, that adhere to a single and clearly defined routing policy, as originally defined in RFC 1771.[2] The newer definition in RFC 1930 came into use because multiple organizations can run BGP using private AS numbers to an ISP that connects all those organizations to the Internet. Even though there may be multiple Autonomous Systems supported by the ISP, the Internet only sees the routing policy of the ISP. That ISP must have an officially registered Autonomous System Number (**ASN**).

A unique ASN is allocated to each AS for use in BGP routing. AS numbers are important because the ASN uniquely identifies each network on the Internet.

Until 2007, AS numbers were defined as 16-bit integers, which allowed for a maximum of 65536 assignments. The Internet Assigned Numbers Authority (IANA) has designated AS numbers 64512 through 65534 to be used for private purposes. The ASNs 0, 59392–64511, and 65535 are reserved by the IANA and should not be used in any routing environment. ASN 0 may be used to label non-routed networks. All other ASNs (1–54271) are subject to assignment by IANA, and, as of September 9, 2008, only 49152–54271 remained unassigned. RFC 4893 introduced 32-bit AS numbers, which IANA has begun to allocate. These numbers are written either as simple integers, or in the form *x.y*, where *x* and *y* are 16-bit numbers. Numbers of the form *0.y* are exactly the old 16-bit AS numbers, *1.y* numbers and 65535.65535 are reserved, and the remainder of the space is available for allocation.[3] The accepted textual representation of Autonomous System Numbers is defined in RFC 5396.[4]

The number of unique autonomous networks in the routing system of the Internet exceeded 5000 in 1999, 30000 in late 2008, and 35000 in mid 2010.[5]

Assignment

AS numbers are assigned in blocks by the Internet Assigned Numbers Authority (IANA) to Regional Internet Registries (RIRs). The appropriate RIR then assigns AS numbers to entities within its designated area from the block assigned by the IANA. Entities wishing to receive an ASN must complete the application process of their local RIR and be approved before being assigned an ASN. Current IANA ASN assignments can be found on the IANA website.[3]

Types

Autonomous Systems can be grouped into three categories, depending on their connectivity and operating policy.

A *multihomed* Autonomous System is an AS that maintains connections to more than one other AS. This allows the AS to remain connected to the Internet in the event of a complete failure of one of their connections. However, this type of AS would not allow traffic from one AS to pass through on its way to another AS.

A *stub* Autonomous System refers to an AS that is connected to only one other AS. This may be an apparent waste of an AS number if the network's routing policy is the same as its upstream AS's. However, the stub AS may in fact have peering with other Autonomous Systems that is not reflected in public route-view servers. Specific examples

include private interconnections in the financial and transportation sectors.

A *transit* Autonomous System is an AS that provides connections through itself to other networks. That is, network A can use network B, the transit AS, to connect to network C. ISPs are always transit ASs, because they provide connections from one network to another. The ISP is considered to be 'selling transit service' to the customer network, thus the term *transit AS*.

References

[1] RFC 1930, Section 3
[2] RFC 1771, original definition (now obsolete) of the Border Gateway Protocol (BGP)
[3] Autonomous System Numbers (http://www.iana.org/assignments/as-numbers)
[4] RFC 5396, *Textual Representation of Autonomous System (AS) Numbers*, G. Huston, G. Michaelson, The Internet Society (December 2008)
[5] Tony Bates, Philip Smith, Geoff Huston. "CIDR report" (http://www.cidr-report.org/as2.0/). . Retrieved 2010-09-17.

External links

- ASN Resource Guide (http://www.apnic.net/services/asn_guide.html)
- AS Number plugin for Mozilla Firefox (http://www.asnumber.networx.ch)
- CIDR and ASN assignment report update continuously (http://www.cidr-report.org/as2.0/)
- Exploring Autonomous System Numbers (http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-1/autonomous_system_numbers.html)
- 32-bit ASN presentation at Doha, Qatar in 11-2007 (http://www.menog.net/meetings/menog2/presentations/philip-smith-32bit-asn.pdf)
- ASN Searching and Tracing by Fixed Orbit (http://www.fixedorbit.com/search.htm)
- ASN Lookup (http://networktools.nl/as-info)
- ashunt, an AS traceroute utility (part of netsniff-ng) (http://www.netsniff-ng.org/)
- Autonomous System Whois Lookup (http://ipduh.com/ip/whois/as/)

Best effort delivery

Best effort delivery describes a network service in which the network does not provide any guarantees that data is delivered or that a user is given a guaranteed quality of service level or a certain priority. In a **best effort network** all users obtain best effort service, meaning that they obtain unspecified variable bit rate and delivery time, depending on the current traffic load.

The postal service delivers letters using a best effort delivery approach. The delivery of a certain letter is not scheduled in advance - no resources are preallocated in the post office. The mailman will make his "best effort" to try to deliver a message, but the delivery may be delayed if too many letters arrive to a postal office suddenly. The sender is not informed if a letter has been delivered successfully.

Conventional telephone networks are not based on best-effort communication, but on circuit switching. During the connection phase of a new call, resources are reserved in the telephone exchanges, or the user is informed that the call is blocked due to lack of free capacity. An ongoing phone call can never be interrupted due to overloading of the network, and is guaranteed constant bandwidth.

Book sources

- *Encyclopedia of Networking & Telecommunications* ISBN 0-07-212005-3

Broadcast address

A **broadcast address** is a logical address at which all devices connected to a multiple-access communications network are enabled to receive datagrams. A message sent to a broadcast address is typically received by all network-attached hosts, rather than by a specific host.

IP networking

In Internet Protocol Version 4 (IPv4) networks, broadcast addresses are formed by a distinguished value for the host identification field, traditionally called the *rest* field, of an IP address.[1] The *all-ones* rest field was established in RFC 919 as the standard broadcast address for networks that support broadcast. This method of using the all-ones address was first proposed by R. Gurwitz and R. Hinden in 1982 (IEN-212).[2] The later introduction of subnets and Classless Inter-Domain Routing changed this slightly, so that the all-ones host address of each subnet is that subnet's broadcast address.

The broadcast address for an IPv4 host can be obtained by performing a bitwise OR operation between the bit complement of the subnet mask and the host's IP address.

Example: For broadcasting a packet to an entire IPv4 subnet using the private IP address space 172.16.0.0/12, which has the subnet mask 255.240.0.0, the broadcast address is 172.16.0.0 | 0.15.255.255 = 172.31.255.255.

A special definition exists for the IP broadcast address `255.255.255.255`. It is the broadcast address of the *zero network* or 0.0.0.0, which in Internet Protocol standards stands for *this network*, i.e. the local network. Transmission to this address is limited by definition, in that it is never forwarded by the routers connecting the local network to other networks.

IP broadcasts are used by BOOTP and DHCP clients to find and send requests to their respective servers.

Internet Protocol Version 6 (IPv6), the next generation Internet Protocol, does not implement the method of broadcast, and therefore does not define broadcast addresses. Instead, IPv6 uses multicast addressing to the *all-hosts* multicast group. No IPv6 protocols are defined to use the all-hosts address, though; instead, they send and receive on

particular link-local multicast addresses. This results in higher efficiency, because network hosts which are not listening for the particular multicast protocol(s) in use are not disturbed or interrupted, as they would be by broadcasts.

Ethernet

Broadcast is possible also on the underlying Data Link Layer in Ethernet networks. Frames are addressed to reach every computer on a given LAN segment if they are addressed to MAC address `FF:FF:FF:FF:FF:FF`. Ethernet frames that contain IP broadcast packages are usually sent to this address.

Ethernet broadcasts are used by Address Resolution Protocol to translate IP addresses to MAC addresses.

IPX networking

The Novell IPX protocol allows broadcast as well. When the network address is specified as `FFFFFFFF`, a packet is sent to all networks available. When the target node address is specified as `FFFFFFFFFFFF`, the packet is intended to be received by all hosts in the network.

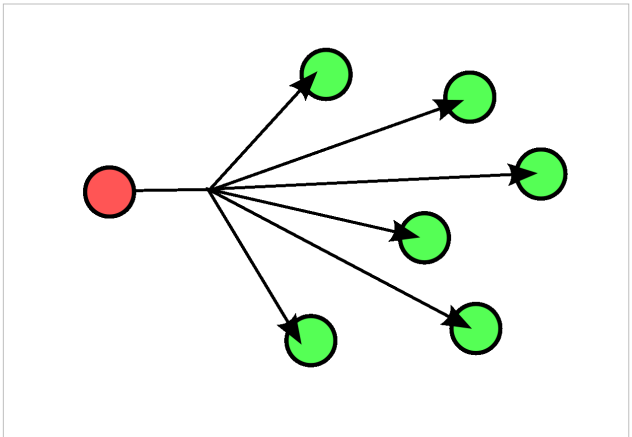
References

[1] RFC 919, *Broadcasting Internet Datagrams*, J. Mogul (October 1984)

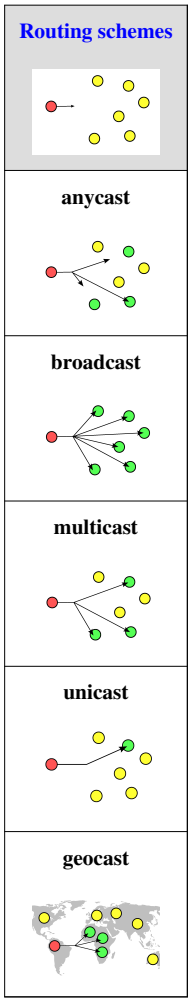
[2] IEN 212 (http://www.rfc-editor.org/ien/ien212.txt), *IP - Local Area Network Addressing Issues*, Robert Gurwitz, Robert Hinden, Bolt Beranek and Newman (BBN) (September 1982)

Broadcasting (computing)

In telecommunication and information theory, **broadcasting** refers to a method of transferring a message to all recipients simultaneously. Broadcasting can be performed as a high level operation in a program, for example broadcasting Message Passing Interface, or it may be a low level networking operation, for example broadcasting on Ethernet.



Overview



In computer networking, **broadcasting** refers to transmitting a packet that will be received by every device on the network[1] . In practice, the scope of the broadcast is limited to a broadcast domain. Broadcast a message is in contrast to unicast addressing in which a host sends datagrams to another single host identified by a unique IP address.

Not all network technologies support broadcast addressing; for example, neither X.25 nor frame relay have broadcast capability, nor is there any form of Internet-wide broadcast. Broadcasting is largely confined to local area network (LAN) technologies, most notably Ethernet and token ring, where the performance impact of broadcasting is not as large as it would be in a wide area network.

The successor to Internet Protocol Version 4 (IPv4), IPv6 also does not implement the broadcast method to prevent disturbing all nodes in a network when only a few may be interested in a particular service. Instead it relies on multicast addressing a conceptually similar *one-to-many* routing methodology. However, multicasting limits the pool of receivers to those that join a specific multicast receiver group.

Both Ethernet and IPv4 use an all-ones broadcast address to indicate a broadcast packet. Token Ring uses a special value in the IEEE 802.2 control field.

Broadcasting may be abused to perform a DoS-attack. The attacker sends fake ping request with the source IP-address of the victim computer. The victim computer is flooded by the replies from all computers in the domain.

References

[1] Andrew Tanenbaum (2003). *Computer Networks*. p. 368. ISBN 0130661023.

External links

- Encyclopedia Britannica entry *broadcast network* (http://www.britannica.com/EBchecked/topic/80540/broadcast-network)
- Network Broadcasting and Multicast (http://www.comptechdoc.org/independent/networking/guide/netbroadcasting.html)

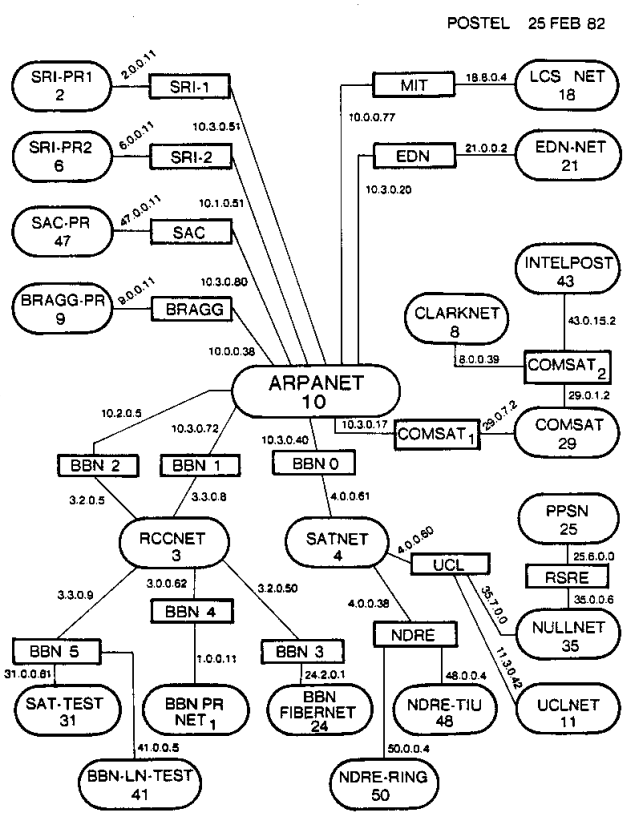
Classful network

A **classful network** is a network addressing architecture used in the Internet from 1981 until the introduction of Classless Inter-Domain Routing in 1993. The method divides the address space for Internet Protocol Version 4 (IPv4) into five address classes. Each class, coded in the first four bits of the address, defines either a different network size, i.e. number of hosts for unicast addresses (classes A, B, C), or a multicast network (class D). The fifth class (E) address range is reserved for future or experimental purposes.

Since its discontinuation, remnants of classful network concepts remain in practice only in limited scope in the default configuration parameters of some network software and hardware components (e.g., default subnet mask), but the terms are often still used in general discussions of network structure among network administrators.

Background

Originally, a 32-bit IPv4 address was logically subdivided into the *network number* field, the most-significant 8 bits of an address, which specified the particular network a host was attached to, and the local address, also called *rest field* (the rest of the address), which uniquely identifies a host connected to that network. This format was sufficient at a time when only a few large networks existed, such as the ARPANET which was assigned the network number 10, and before the wide proliferation of local area networks (LANs). As a consequence of this architecture, the address space supported only a low number (254) of independent networks, and it became clear very early on that this would not be enough.



Map of the prototype Internet in 1982, showing 8-bit-numbered networks (ovals) only, interconnected by routers (rectangles).

Introduction of address classes

Expansion of the network had to ensure compatibility with the existing address space and

the Internet Protocol (IP) packet structure, and avoid the renumbering of the existing networks. The solution was to expand the definition of the network number field to include more bits, allowing more networks to be designated, each potentially having fewer hosts. All existing network numbers at the time were smaller than 64, they only used the 6 least-significant bits of the network number field. Thus it was possible to use the most-significant bits of an address to introduce a set of address classes, while preserving the existing network numbers in the first of these classes.

The new addressing architecture was introduced by RFC 791 in 1981 as a part of the specification of the Internet Protocol.[1] It divided the address space into primarily three address formats, henceforth called address *classes*, and left a fourth range reserved to be defined later.

The first class, designated as *Class A*, contained all addresses in which the most significant bit is zero. The network number for this class is given by the next 7 bits, therefore accommodating 128 networks in total, including the zero network, and including the existing IP networks already allocated. A *Class B* network was a network in which all addresses had the two most-significant bits set to 1 and 0. For these networks, the network address was given by the next 14 bits of the address, thus leaving 16 bits for numbering host on the network for a total of 65536 addresses per network. *Class C* was defined with the 3 high-order bits set to 1, 1, and 0, and designating the next 21 bits to number the networks, leaving each network with 256 local addresses.

The leading bit sequence *111* designated an "*escape to extended addressing mode*",[1] which was later subdivided in to Class D (*1110*) for multicast addressing, while leaving as reserved for future use the *1111* block designated as Class E.

This addressing scheme is illustrated in the following table:

Class	**Leading bits**	**Size of *network number* bit field**	**Size of *rest* bit field**	**Number of networks**	**Addresses per network**	**Start address**	**End address**
Class A	0	8	24	128 (2^7)	16,777,216 (2^{24})	0.0.0.0	127.255.255.255
Class B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	128.0.0.0	191.255.255.255
Class C	110	24	8	2,097,152 (2^{21})	256 (2^8)	192.0.0.0	223.255.255.255
Class D (multicast)	1110	not defined	not defined	not defined	not defined	224.0.0.0	239.255.255.255
Class E (reserved)	1111	not defined	not defined	not defined	not defined	240.0.0.0	255.255.255.255

The number of addresses usable for addressing specific hosts in each network is always 2^N - 2 (where N is the number of rest field bits, and the subtraction of 2 adjusts for the use of the all-bits-zero host portion for network address and the all-bits-one host portion as a broadcast address. Thus, for a Class C address with 8 bits available in the host field, the number of hosts is 254.

Today, IP addresses are associated with a subnet mask. This was not required in a classful network because the mask was implicitly derived from the IP address itself. Any network device would inspect the first few bits of the IP address to determine the class of the address.

Bit-wise representation

In the following table:

- *n* indicates a binary slot used for network ID.
- *H* indicates a binary slot used for host ID.
- *X* indicates a binary slot (without specified purpose)

```
Class A
  0.  0.  0.  0 = 00000000.00000000.00000000.00000000
127.255.255.255 = 01111111.11111111.11111111.11111111
                  0nnnnnnn.HHHHHHHH.HHHHHHHH.HHHHHHHH

Class B
128.  0.  0.  0 = 10000000.00000000.00000000.00000000
191.255.255.255 = 10111111.11111111.11111111.11111111
                  10nnnnnn.nnnnnnnn.HHHHHHHH.HHHHHHHH

Class C
192.  0.  0.  0 = 11000000.00000000.00000000.00000000
223.255.255.255 = 11011111.11111111.11111111.11111111
                  110nnnnn.nnnnnnnn.nnnnnnnn.HHHHHHHH

Class D
224.  0.  0.  0 = 11100000.00000000.00000000.00000000
239.255.255.255 = 11101111.11111111.11111111.11111111
                  1110XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX

Class E
240.  0.  0.  0 = 11110000.00000000.00000000.00000000
255.255.255.255 = 11111111.11111111.11111111.11111111
                  1111XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX
```

The replacement of classes

The first architecture change extended the addressing capability in the Internet, but did not prevent IP address shortage. The problem was that many sites needed larger address blocks than a Class C network provided, and therefore they received a Class B block, which was in most cases much larger than required. In the rapid growth of the Internet, the pool of unassigned Class B addresses (2^{14}, or about 16,000) was rapidly being depleted. Classful networking was replaced by Classless Inter-Domain Routing (CIDR), starting in 1993 with the specification of RFC 1518 and RFC 1519, to attempt to solve this problem.

Early allocations of IP addresses by the Internet Assigned Numbers Authority (IANA) were in some cases not made efficiently, which contributed to the problem. However, the commonly held notion that some American organizations unfairly or unnecessarily received Class A networks is wrong; most such allocations date to the period before the introduction of address classes, when the only address blocks available were what later became known as Class A networks.[2]

References

[1] RFC 791, *Internet Protocol*, Information Sciences Institute (September 1981)
[2] IEN46, *A proposal for addressing and routing in the Internet* (http://www.potaroo.net/ietf/ien/ien46.txt), David D. Clark, June 1978

External links

- IANA, Current IPv4 /8 delegations (http://www.iana.org/assignments/ipv4-address-space)
- Overview of IP addressing, both classless and classful (http://oreilly.com/catalog/coreprot/chapter/appb.html)

Classless Inter-Domain Routing

Classless Inter-Domain Routing (**CIDR**) is a method for allocating IP addresses and routing Internet Protocol packets. The Internet Engineering Task Force introduced CIDR in 1993 to replace the previous addressing architecture of classful network design in the Internet. Their goal was to slow the growth of routing tables on routers across the Internet, and to help slow the rapid exhaustion of IPv4 addresses.[1] [2]

IP addresses are described as consisting of two groups of bits in the address: the most significant part is the *network address* which identifies a whole network or subnet and the least significant portion is the *host identifier*, which specifies a particular interface of a host on that network. This division is used as the basis of traffic routing between IP networks and for address allocation policies. Classful network design for IPv4 sized the network address as one or more 8-bit groups, resulting in the blocks of Class A, B, or C addresses. Classless Inter-Domain Routing allocates address space to Internet service providers and end users on any address bit boundary, instead of on 8-bit segments. In IPv6, however, the interface identifier has a fixed size of 64 bits by convention, and smaller subnets are never allocated to end users.

CIDR notation is a syntax of specifying IP addresses and their associated routing prefix. It appends to the address a slash character and the decimal number of leading bits of the routing prefix, e.g., 192.168.0.0/16 for IPv4, and 2001:db8::/32 for IPv6.

Background

During the first decade of the modern Internet after the invention of the Domain Name System (DNS) it became apparent that the devised system based on the classful network scheme of allocating the IP address space and the routing of IP packets was not scalable.[3]

To alleviate the shortcomings, the Internet Engineering Task Force published in 1993 a new set of standards, RFC 1518 and RFC 1519, to define a new concept of allocation of IP address blocks and new methods of routing IPv4 packets. A new version of the specification was published as RFC 4632 in 2006.[4]

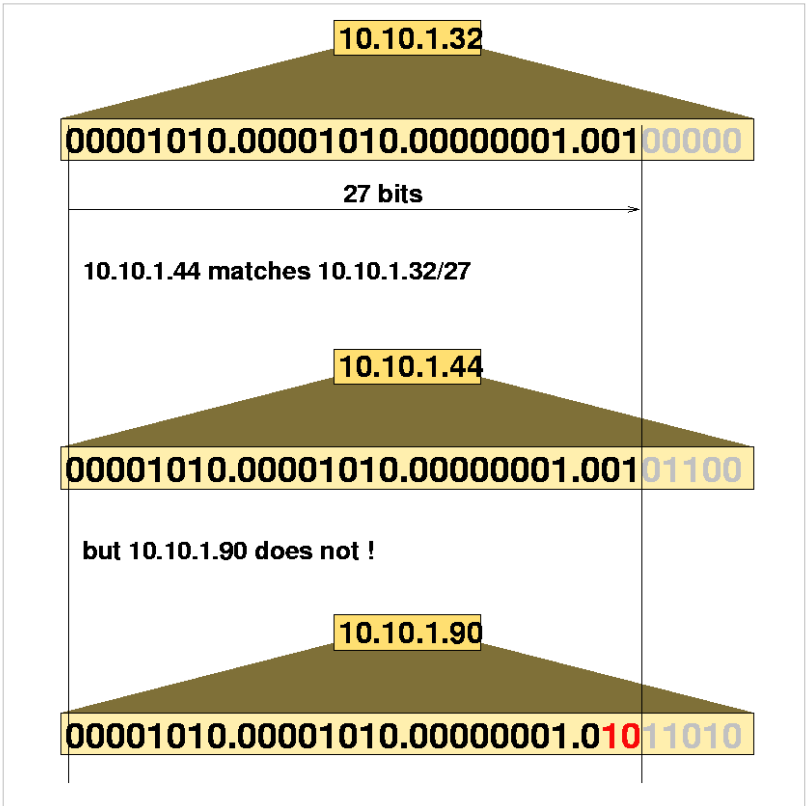
An IP address is interpreted as composed of two parts: a network-identifying prefix followed by a host identifier within that network. In the previous classful network architecture of Internet Protocol Version 4, IP address allocations were based on the bit boundaries of the four octets of an IP address. An address was considered to be the combination of an 8, 16, or 24-bit network prefix along with a 24, 16, or 8-bit individual or node address. Thus, the smallest allocation and routing block contained only 256 addresses—too small for most enterprises, and the next larger block contained 65,536 addresses—too large to be used efficiently by even large organizations. This led to inefficiencies in address use as well as routing because the large number of allocated small (class-C) networks with individual route announcements, being geographically dispersed with little opportunity for route aggregation, created heavy demand on routing equipment.

As the experimental TCP/IP network expanded into the Internet during the 1980s, the need for more flexible addressing schemes became increasingly apparent. This led to the successive development of subnetting and CIDR. Because the old class distinctions are ignored, the new system was called *classless routing*. It is supported by modern routing protocols, such as RIP-2, EIGRP, IS-IS and OSPF. This led to the original system being called, by back-formation, classful routing.

Classless Inter-Domain Routing is based on *variable-length subnet masking* (VLSM), which allows a network to be divided into different-sized subnets. CIDR avoids wasting IP addresses. Variable-length subnet masks are mentioned in RFC 950. [5]

CIDR encompasses several concepts. It is based on the VLSM technique with effective qualities of specifying arbitrary-length prefixes. Addresses are represented in CIDR notation, in which an address or routing prefix is written with a suffix indicating the number of bits in the address, such as 192.168.0.0/16. CIDR introduced an administrative process of allocating address blocks to organizations based on their actual and short-term projected needs. The aggregation of multiple contiguous prefixes resulted in supernets in the larger Internet, which when ever possible are advertised as aggregates, thus reducing the number of entries in the global routing table.

CIDR blocks



CIDR is principally a bitwise, prefix-based standard for the interpretation of IP addresses. It facilitates routing by allowing blocks of addresses to be grouped into single routing table entries. These groups, commonly called CIDR blocks, share an initial sequence of bits in the binary representation of their IP addresses. IPv4 CIDR blocks are identified using a syntax similar to that of IPv4 addresses: a four-part dotted-decimal address, followed by a slash, then a number from 0 to 32: A.B.C.D/N. The dotted decimal portion is interpreted, like an IPv4 address, as a 32-bit binary number that has been broken into four octets. The number following the slash is the prefix length, the number of shared initial bits, counting from the most-significant bit of the address. When emphasizing only the size of a network, the address portion of the notation is usually omitted. Thus, a /20 is a CIDR block with an unspecified 20-bit prefix.

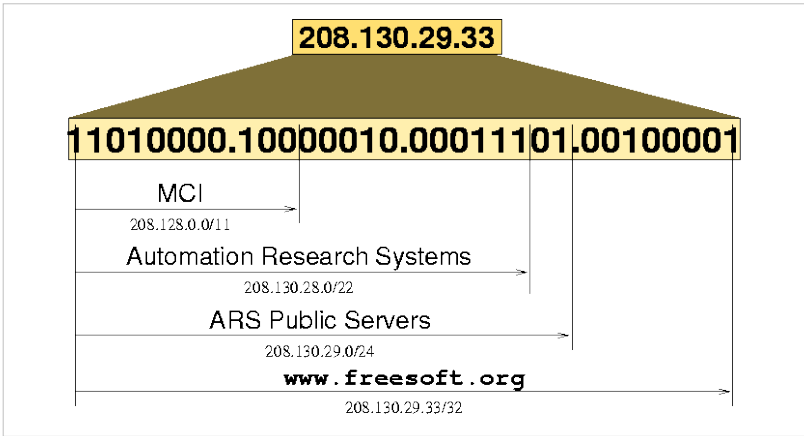
An IP address is part of a CIDR block, and is said to match the CIDR prefix if the initial N bits of the address and the CIDR prefix are the same. Thus, understanding CIDR requires that IP address be visualized in binary. Since the length of an IPv4 address has 32 bits, an N-bit CIDR prefix leaves 32-N bits unmatched, meaning that 2^{32-N} IPv4 addresses match a given N-bit CIDR prefix. Shorter CIDR prefixes match more addresses, while longer prefixes match fewer. An address can match multiple CIDR prefixes of different lengths.

CIDR is also used for IPv6 addresses and the syntax semantic is identical. A prefix length can range from 0 to 128, due to the larger number of bits in the address, however, by convention a subnet on broadcast MAC layer networks

always has 64-bit host identifiers. Larger prefixes are rarely used even on point-to-point links.

Assignment of CIDR blocks

The Internet Assigned Numbers Authority (IANA) issues to regional Internet registries (RIRs) large, short-prefix CIDR blocks. For example, 62.0.0.0/8, with over sixteen million addresses, is administered by RIPE NCC, the European RIR. The RIRs, each responsible for a single, large, geographic area, such as Europe or North America, then subdivide these blocks into smaller blocks and issue them to local Internet registries. This subdividing process can be repeated several times at different levels of delegation. End user networks receive subnets sized according to the size of their network and projected short term need. Networks served by a single ISP are encouraged by IETF recommendations to obtain IP address space directly from their ISP. Networks served by multiple ISPs, on the other hand, may obtain provider-independent address space directly from the appropriate RIR.



For example, in the late 1990s, the IP address 208.130.29.33 (since reassigned) was used by www.freesoft.org. An analysis of this address identified three CIDR prefixes. 208.128.0.0/11, a large CIDR block containing over 2 million addresses, had been assigned by ARIN (the North American RIR) to MCI. Automation Research Systems, a Virginia VAR, leased an Internet connection from MCI and was assigned the 208.130.28.0/22 block, capable of addressing just over 1000 devices. ARS used a /24 block for its publicly accessible servers, of which 208.130.29.33 was one.

All of these CIDR prefixes would be used, at different locations in the network. Outside of MCI's network, the 208.128.0.0/11 prefix would be used to direct to MCI traffic bound not only for 208.130.29.33, but also for any of the roughly two million IP addresses with the same initial 11 bits. Within MCI's network, 208.130.28.0/22 would become visible, directing traffic to the leased line serving ARS. Only within the ARS corporate network would the 208.130.29.0/24 prefix have been used.

Subnet masks

A subnet mask is a bitmask that encodes the prefix length in quad-dotted notation: 32 bits, starting with a number of 1 bits equal to the prefix length, ending with 0 bits, and encoded in four-part dotted-decimal format. A subnet mask encodes the same information as a prefix length, but predates the advent of CIDR. However, in CIDR notation, the prefix bits are always contiguous, whereas subnet masks may specify non-contiguous bits. However, this has no practical advantage for increasing efficiency.

Prefix aggregation

CIDR provides the possibility of fine-grained *routing prefix aggregation*, also known as *supernetting* or *route summarization*. For example, sixteen contiguous `/24` networks can be aggregated and advertised to a larger network as a single `/20` route, if the first 20 bits of their network addresses match. Two aligned contiguous `/20`s may then be aggregated to a `/19`, and so forth. This allows a significant reduction in the number of routes that have to be advertised.

IPv4 CIDR Blocks

IPv4 CIDR

IP/CIDR	Δ to last IP addr	Mask	Hosts (*)	Class	Notes
a.b.c.d/**32**	+0.0.0.0	255.255.255.255	1	1/256 C	
a.b.c.d/**31**	+0.0.0.1	255.255.255.254	2	1/128 C	d = 0 ... (2n) ... 254
a.b.c.d/**30**	+0.0.0.3	255.255.255.252	4	1/64 C	d = 0 ... (4n) ... 252
a.b.c.d/**29**	+0.0.0.7	255.255.255.248	8	1/32 C	d = 0 ... (8n) ... 248
a.b.c.d/**28**	+0.0.0.15	255.255.255.240	16	1/16 C	d = 0 ... (16n) ... 240
a.b.c.d/**27**	+0.0.0.31	255.255.255.224	32	1/8 C	d = 0 ... (32n) ... 224
a.b.c.d/**26**	+0.0.0.63	255.255.255.192	64	1/4 C	d = 0, 64, 128, 192
a.b.c.d/**25**	+0.0.0.127	255.255.255.128	128	1/2 C	d = 0, 128
a.b.c.0/**24**	+0.0.0.255	255.255.255.000	256	1 C	
a.b.c.0/**23**	+0.0.1.255	255.255.254.000	512	2 C	c = 0 ... (2n) ... 254
a.b.c.0/**22**	+0.0.3.255	255.255.252.000	1,024	4 C	c = 0 ... (4n) ... 252
a.b.c.0/**21**	+0.0.7.255	255.255.248.000	2,048	8 C	c = 0 ... (8n) ... 248
a.b.c.0/**20**	+0.0.15.255	255.255.240.000	4,096	16 C	c = 0 ... (16n) ... 240
a.b.c.0/**19**	+0.0.31.255	255.255.224.000	8,192	32 C	c = 0 ... (32n) ... 224
a.b.c.0/**18**	+0.0.63.255	255.255.192.000	16,384	64 C	c = 0, 64, 128, 192
a.b.c.0/**17**	+0.0.127.255	255.255.128.000	32,768	128 C	c = 0, 128
a.b.0.0/**16**	+0.0.255.255	255.255.000.000	65,536	256 C = 1 B	
a.b.0.0/**15**	+0.1.255.255	255.254.000.000	131,072	2 B	b = 0 ... (2n) ... 254
a.b.0.0/**14**	+0.3.255.255	255.252.000.000	262,144	4 B	b = 0 ... (4n) ... 252
a.b.0.0/**13**	+0.7.255.255	255.248.000.000	524,288	8 B	b = 0 ... (8n) ... 248
a.b.0.0/**12**	+0.15.255.255	255.240.000.000	1,048,576	16 B	b = 0 ... (16n) ... 240
a.b.0.0/**11**	+0.31.255.255	255.224.000.000	2,097,152	32 B	b = 0 ... (32n) ... 224
a.b.0.0/**10**	+0.63.255.255	255.192.000.000	4,194,304	64 B	b = 0, 64, 128, 192
a.b.0.0/**9**	+0.127.255.255	255.128.000.000	8,388,608	128 B	b = 0, 128
a.0.0.0/**8**	+0.255.255.255	255.000.000.000	16,777,216	256 B = 1 A	
a.0.0.0/**7**	+1.255.255.255	254.000.000.000	33,554,432	2 A	a = 0 ... (2n) ... 254
a.0.0.0/**6**	+3.255.255.255	252.000.000.000	67,108,864	4 A	a = 0 ... (4n) ... 252
a.0.0.0/**5**	+7.255.255.255	248.000.000.000	134,217,728	8 A	a = 0 ... (8n) ... 248
a.0.0.0/**4**	+15.255.255.255	240.000.000.000	268,435,456	16 A	a = 0 ... (16n) ... 240

a.0.0.0/**3**	+31.255.255.255	224.000.000.000	536,870,912	32 A	a = 0 ... (32n) ... 224
a.0.0.0/**2**	+63.255.255.255	192.000.000.000	1,073,741,824	64 A	a = 0, 64, 128, 192
a.0.0.0/**1**	+127.255.255.255	128.000.000.000	2,147,483,648	128 A	a = 0, 128
0.0.0.0/**0**	+255.255.255.255	000.000.000.000	4,294,967,296	256 A	

(*) For routed subnets bigger than `/31` or `/32`, two reserved addresses need to be subtracted from the number of available host addresses: the largest address, which is used as the broadcast address, and the smallest address, which is used to identify the network itself. [6] [7] It is also common for the IP gateway for that subnet to use an address, meaning that three addresses would have to be subtracted from the number of hosts that can be used on the subnet.

IPv6 CIDR blocks

IPv6 is classless. However, some worry that if the recommended CIDR blocks allocations/assignments become the default that would be equivalent to using classes[8] . Well, it is extremely difficult to run out of IPv6 addresses but the growing of the routing table is inevitable.

IPv6 CIDR blocks and their recommended uses

```
2001:0db8:0123:4567:89ab:cdef:1234:5678
|||| |||| |||| |||| |||| |||| |||| ||||
|||| |||| |||| |||| |||| |||| |||| |||128   /128 Single end-points and loopback
|||| |||| |||| |||| |||| |||| |||| ||124
|||| |||| |||| |||| |||| |||| |||| |120
|||| |||| |||| |||| |||| |||| |||| 116
|||| |||| |||| |||| |||| |||| |||112
|||| |||| |||| |||| |||| |||| ||108
|||| |||| |||| |||| |||| |||| |104
|||| |||| |||| |||| |||| |||| 100
|||| |||| |||| |||| |||| |||96
|||| |||| |||| |||| |||| ||92
|||| |||| |||| |||| |||| |88
|||| |||| |||| |||| |||| 84
|||| |||| |||| |||| |||80
|||| |||| |||| |||| ||76
|||| |||| |||| |||| |72
|||| |||| |||| |||| 68
|||| |||| |||| |||64   /64 Single End-user LAN Subnet (default prefix size for SLAAC) [8]
|||| |||| |||| ||60   /60 Some (very limited) 6rd deployments
|||| |||| |||| |56   /56 Minimal end sites assignment (e.g. Home network) [8]
|||| |||| |||| 52
|||| |||| |||48   /48 Typical assignment for larger sites [8]
|||| |||| ||44
|||| |||| |40
|||| |||| 36   /36 possible future Local Internet registry extra-small allocations
|||| |||32   /32 Local Internet registry minimum allocations
|||| ||28   /28 Local Internet registry medium allocations
|||| |24   /24 Local Internet registry large allocations
|||| 20   /20 Local Internet registry extra large allocations
|||16
```

```
||12   /12 Regional Internet Registry allocations from IANA[9]
|8
4
```

(*)The number of hosts in a IPv6 CIDR block is equal to 2 ^ (128 - prefix).

References

[1] RFC 1518, *An Architecture for IP Address Allocation with CIDR*, Y. Rekhter, T. Li (September 1993)

[2] RFC 1519, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, V. Fuller, T. Li, J. Yu, K. Varadhan (September 1993)

[3] RFC 1517, *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)*, R. Hinden, Ed., (September 1993)

[4] RFC 4632, *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, V. Fuller, T. Li (August 2006)

[5] RFC 950, *Internet Standard Subnetting Procedure*, J. Mogul, J. Postel, Eds. (August 1985), Section 2.1

[6] RFC 922, *Broadcasting Internet Datagrams in the Presence of Subnets*, J. Mogul (Ed.), October 1984, Section 7.

[7] RFC 1812, *Requirements for IP Version 4 Routers*, F. Baker (Ed.), June 1995, Section 4.2.3.1

[8] RFC 6177 IPv6 Address Assignment to End Sites

[9] IANA IPv6 unicast address assignments (http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml)

External links

- RFC 1518, *An Architecture for IP Address Allocation with CIDR*
- RFC 2317, *Classless IN-ADDR.ARPA delegation*
- RFC 3021, *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*
- RFC 4291, *IP Version 6 Addressing Architecture*
- CIDR Report (http://www.cidr-report.org/) (updated daily)
- CIDR FAQ (http://www.faqs.org/rfcs/rfc1817.html)
- CIDR Calculator (http://ipduh.com/ip/cidr)

End-to-end principle

The **end-to-end principle** is a classic design principle of computer networking which states that application specific functions ought to reside in the *end hosts* of a network rather than in *intermediary nodes*, provided they can be implemented "completely and correctly" in the end hosts. First explicitly articulated in a 1981 conference paper by Saltzer, Reed, and Clark,[1] [2] it has inspired and informed many subsequent debates on the proper distribution of functions in the Internet and communication networks more generally.

Going back to Paul Baran's work on obtaining reliability from unreliable parts in the early 1960s, the basic intuition behind the original principle is that the payoffs from adding functions to the *network* quickly diminish, especially in those cases where the *end hosts* will have to implement functions for reasons of "completeness and correctness", any efforts of the network notwithstanding. The canonical example for the end-to-end principle is that of arbitrarily reliable data transfer between two end hosts. However, the principle has also, for example, shown to apply to the question of buffer sizes in intermediary nodes.

A common misperception about the end-to-end principle is that it implies network neutrality or a "dumb" network. First, there are functions orthogonal to application specifics or which simply cannot be implemented "completely and correctly" in the network ends; and, second, there may be good reasons to offload functions from end hosts to intermediary nodes, for performance reasons or when these are better equipped to perform the functions or part thereof.

History

The concept and research of end-to-end connectivity and network intelligence at the end nodes reaches back to packet-switching networks in the 1970s, cf. CYCLADES. A 1981 academic paper entitled *End-to-end arguments in system design*[1] by Jerome H. Saltzer, David P. Reed, and David D. Clark, argued that reliable systems tend to require end-to-end processing to operate correctly, in addition to any processing in the intermediate system. They pointed out that most features in the lowest level of a communications system have costs for all higher-layer clients, even if those clients do not need the features, and are redundant if the clients have to reimplement the features on an end-to-end basis.

This is sometimes said to lead to a model of a *dumb, minimal network* with smart terminals, a completely different model from the previous paradigm of the smart network with dumb terminals.

In 1995, the Federal Networking Council adopted a resolution defining the Internet as a “global information system” that is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons; is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and provides, uses or makes accessible, either publicly or privately, high level services layered on this communications and related infrastructure.[3]

Limitations

The rapid expansion of the Internet and the resulting IPv4 address exhaustion problem have forced some design changes in the originally conceived architecture of the IP address space in terms of address assignment and routing technologies. Technologies invented to alleviate the exhaustion problem have introduced elements, such as network address translation, that do not abide by the end-to-end principle. Without this property, some network protocols require the specific support of network elements during traversal. This impediment hinders deployment of many new, often interactive, applications, including security (IPsec), migration to IPv6 (tunneling IPv6 in IPv4), peer-to-peer applications, and networked games.

Sometimes end-to-end connectivity is deliberately broken, erroneously, as a means of implementation of network security, as using address translation also limits the routing scope, which means that computers behind NAT cannot be addressed directly from untrusted zones. However, consensus among security experts shows that this does not provide proper security features and in fact may impede implementation of proper techniques.

Such implementation trends divide Internet users into those who have "real" Internet connectivity and those who are restricted to use applications that only use outbound network connections.

Examples

In the Internet Protocol Suite, the Internet Protocol is a simple ("dumb"), stateless protocol that moves datagrams across the network, and TCP is a smart transport protocol providing error detection, retransmission, congestion control, and flow control end-to-end. The network itself (the routers) needs only to support the simple, lightweight IP; the endpoints run the heavier TCP on top of it when needed.

A second canonical example is that of file transfer. Every reliable file transfer protocol and file transfer program should contain a checksum, which is validated only after everything has been successfully stored on disk. Disk errors, router errors, and file transfer software errors make an end-to-end checksum necessary. Therefore, there is a limit to how secure TCP checksum should be, because it has to be reimplemented for any robust end-to-end application to be secure.

A third example (not from the original paper) is the EtherType field of Ethernet. An Ethernet frame does not attempt to provide interpretation for the 16 bits of type. To add special interpretation to some of these bits would reduce the total number of Ethertypes, hurting the scalability of higher layer protocols, i.e. all higher layer protocols would pay a price for the benefit of just a few. Attempts to add elaborate interpretation (e.g. IEEE 802 SSAP/DSAP) have generally been ignored by most network designs.

Views on the principle

Some view the end-to-end principle as a pragmatic engineering philosophy for network system design that merely prefers putting intelligence towards the end points. It does not forbid intelligence in the network itself if it makes more practical sense.

> from the beginning, the end to end arguments revolved around requirements that could be implemented correctly at the end-points; if implementation inside the network is the only way to accomplish the requirement, then an end to end argument isn't appropriate in the first place.

[4]

Indeed, as noted in RFC 1958 edited by Brian Carpenter in June 1996, entitled “Architectural Principles of the Internet,” “[i]n searching for Internet architectural principles, we must remember that technical change is continuous in the information technology industry. The Internet reflects this. . . .In this environment, some architectural principles inevitably change. Principles that seemed inviolable a few years ago are deprecated tomorrow. The principle of constant change is perhaps the only principle of the Internet that should survive indefinitely.” This is particularly true with respect to the so-called “end-to-end” principle.

As noted by Bob Kahn, co-inventor of the Internet Protocol:

> The original Internet involved three individual networks, namely the ARPANET, the Packet Radio network and the Packet Satellite network, all three of which had been developed with DARPA support. One early consideration that was rejected was to change each of these networks to be able to interpret and route internet packets so that there would be no need for external devices to route the traffic. However, this would have required major changes to all three networks and would have required synchronized changes in all three to accommodate protocol evolutions. Instead, it was decided to create what were called “gateways,” the forerunner of today’s routers, to handle the IP protocol-based networks.

> Reliable packet communication was handled by a combination of factors, but, ultimately, the TCP protocol provided an end-to-end means of reassembly of packet fragments, error checking and acknowledgment back to the source. The resulting fact that no changes were needed in the individual networks was interpreted by some as implying that the Internet design assumed only dumb networks with all the smarts being at the boundaries. Nothing could have been further from the truth. The initial choice of using gateways/routers was purely pragmatic and should imply nothing about how the Internet might operate in the future.

In comments submitted by Patrice Lyons to the United Nations Working Group on Internet Governance (November 4, 2004), entitled "The End-End Principle and the Definition of Internet," on behalf of Bob Kahn's non profit research organization, Corporation for National Research Initiatives (CNRI), it was noted that:

> To argue today that the only stateful elements that may be active in the Internet environment should be located at the edges of the Internet is to ignore the evolution of software and other technologies to provide a host of services throughout the Internet. The layering approach has many advantages and should be retained along with more integrated system architectures; the approach was a practical way of overlaying the Internet architecture over existing networks when it was difficult to coordinate the modification of these networks, if indeed such modifications could have been agreed upon and implemented. For some newer applications, maintaining state information within the network may now be desirable for efficiency if not overall performance effectiveness. In addition, current research efforts may need to draw upon innovative methods to increase security of communications, develop new forms of structuring data, create and deploy dynamic metadata repositories, or real-time authentication of the information itself.

"The End-End Principle and the Definition of Internet" [5]. 2004-11-10. Retrieved 2011-05-16.

Specifically, CNRI proposed that, in the third element of the FNC definition of Internet, after the phrase "high level services layered on", it is advisable to add the following words: "or integrated with", and observed that this point is "directly relevant to the ongoing discussions about the so-called 'end-to-end' principle that is often viewed as essential to an understanding of the Internet". Further, while the end-to-end principle may have been relevant in the environment where the Internet originated, it has not been critical for a number of years going back "at least to the early work on mobile programs, distributed searching, and certain aspects of collaborative computing".

Notes

[1] Saltzer, J.H.; Reed, D.P.; Clark, D.D. (April 1981). "End-To-End Arguments in System Design" (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf). *Proceedings of the 2nd International Conference on Distributed Computing Systems* (IEEE Computer Society): 509–512. .

[2] The 1981 paper was published in ACM's TOCS in an updated version in 1984.

[3] "Definition of "Internet"" (http://www.nitrd.gov/fnc/Internet_res.html). 1995-10-24. . Retrieved 2011-05-16.

[4] Blumenthal, M.S, Clark, D.D., *Rethinking the design of the Internet: The end to end arguments vs. the brave new world*. ACM Transactions on Internet Technology, 1(1), pages 70-109, 2001.

[5] http://www.wgig.org/docs/CNRInovember.pdf

References

External links

- The Rise of the Middle and the Future of End-to-End (http://www.faqs.org/rfcs/rfc3724.html) Reflections on the Evolution of the Internet Architecture (lots of references)
- Active Networking and E2E (http://web.mit.edu/Saltzer/www/publications/endtoend/ANe2ecomment.html)
- The end to end arguments vs. the brave new world (http://hdl.handle.net/1721.1/1519) paper
- End-to-End Arguments in the Internet: Principles, Practices, and Theory (http://www.xn--brwolff-5wa.de/publications/2010-10-PhD-thesis.html) lengthy thesis on the history and shape of the principle
- End-to-End Ethernet Connectivity Fault Management in Metro and Access Networks (http://www.alcatel.com/publications/abstract.jhtml?repositoryItem=tcm:172-288441635) Technology White Paper
- End to End (http://www.cybertelecom.org/notes/end_to_end.htm) E2E Links
- RFC 3724, The Rise of the Middle and the Future of End-to-End Reflections on the Evolution of the Internet Architecture

Holddown

Holddown timer works by having each router start a timer when they first receive information about a network that is unreachable. Until the timer expires, the router will discard any subsequent route messages that indicate the route is in fact reachable. It can solve the case where multiple routers are connected indirectly. The classic example would be three routers configured in a triangle. In this situation, Split horizon and split horizon with poisoned reverse can do nothing.[1]

In other words, a holddown keeps a router from receiving route updates until the network appears to be stable—until either an interface stops changing state (flapping) or a better route is learned.

Holddowns are usually implemented with timers. If the router detects a network unreachable, the timer is started. The router will then wait a preset number of seconds until the network is stable. When the timer expires, the router will then receive its routing updates from other routers. For example in RIP the default holddown timer is set on 180 seconds.

References

[1] The TCP/IP Guide-RIP Special Features For Resolving RIP Algorithm Problems

Hop-by-hop transport

Hop-by-hop transport is a principle of controlling the flow of data in a network. With hop-by-hop transport, chunks of data are forwarded from node to node in a store-and-forward manner.

As hop-by-hop transport involves not only the source and destination node, but rather some or all of the intermediate nodes as well, it allows data to be forwarded even if the path between source and destination is not permanently connected during communication.

However, the End-to-end principle claims that transport control should be implemented end-to-end *unless* implementing hop-by-hop transport achieves considerably better performance. Moreover, hop-by-hop transport requires per-flow state information at intermediate nodes, which limits its scalability. This is one of the reasons why almost all communication today is controlled by end-to-end transport protocols such as TCP.

Current research in the area of sparse mobile networks is considering hop-by-hop transport for application scenarios where end-to-end connectivity is only available intermittently, as under such conditions, hop-by-hop transport can achieve substantial performance gains.

External links

- On Leveraging Partial Paths in Partially-Connected Networks [1]
- End-to-end vs. hop-by-hop transport under intermittent connectivity [2]
- The Transport Layer Revisited [3]
- CAT: Contention Aware Transport Protocol for IEEE 802.11 MANETs [4]

References

[1] http://dx.doi.org/10.1109/INFCOM.2009.5061906
[2] http://portal.acm.org/citation.cfm?id=1365589
[3] http://dx.doi.org/10.1109/COMSWA.2007.382568
[4] http://dx.doi.org/10.1109/VETECS.2006.1682879

Host (network)

A **network host** is a computer connected to a computer network. A network host may offer information resources, services, and applications to users or other nodes on the network. A network host is a network node that is assigned a network layer host address.

Computers participating in networks that use the Internet Protocol Suite may also be called IP hosts. Specifically, computers participating in the Internet are called *Internet hosts*, sometimes Internet nodes. Internet hosts and other IP hosts have one or more IP addresses assigned to their network interfaces. The addresses are configured either manually by an administrator, automatically at start-up by means of the Dynamic Host Configuration Protocol (DHCP), or by stateless address autoconfiguration methods.

Every network host is a physical network node (i.e. a network device), but not every physical network node is a host. Network devices such as modems, hubs and network switches are not assigned host addresses (except sometimes for administrative purposes), and are consequently not considered as network hosts. Devices such as network printers and hardware routers have IP addresses, but since they are not general-purpose computers, they are sometimes not considered as hosts.

Network hosts that participate in applications that use the client-server model of computing, are classified as server or client systems. Network hosts may also function as nodes in peer-to-peer applications, in which all nodes share and consume resources in an equipotent manner.

Server and Host

Host	Server
Always a physical node	Can be a physical node or a software program
Can run both server and client programs	Installed on a host
Provides specific services	Provides specific services
Serves multiple users and devices	Serves only clients

Origin of the concept

In operating systems, the term terminal host traditionally denotes a multi-user computer or software providing services to computer terminals, or a computer that provides services to smaller or less capable devices,[1] such as a main frame computers serving teletype terminals or video terminals. Other examples are a telnet host (a telnet server) and an xhost (X Window client).

The term "Internet host" or just "host" is used in a number of Request for Comments (RFC) documents that define the Internet and its predecessor, the ARPANET. While the ARPANET was developed, computers connected to the network were typically main frame computer systems that could be accessed from terminals connected via serial ports. Since these dumb terminals did not host software or perform computations themselves, they were not considered hosts. The terminals were connected to the terminal hosts through serial interfaces and perhaps circuit switched networks, but not connected to any IP based network, and were not assigned IP addresses. Today's IP hosts may, however, lack ability to serve as terminal hosts.

RFC 871 defines a *host*, as a general-purpose computer system connected to a communications network for "... *the purpose of achieving resource sharing amongst the participating operating systems,...*".[2]

References

[1] E. Garrison Walters, essential guide to computing (http://books.google.com/books?id=AwrQsOW5SsQC&pg=PA149&dq=terminal+"host+"+a+multi-user&hl=sv&ei=1jwCTfudOcbysgb2xuCLCg&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCkQ6AEwAA#v=onepage&q=multi-user host&f=falseThe), page 149, Prentice Hall PTR, 2001.

[2] RFC 871, *A Perspective on the ARPANET Reference Model*, M.A. Padlipsky (September 1982)

ICMP Redirect Message

The ICMP type 5 contains a redirect message to send data packets on an alternative route. ICMP Redirect is a mechanism for routers to convey routing information to hosts. The **Redirect Message** is an ICMP message which informs a host to update its routing information (to send packets on an alternate route). If a host tries to send data through a router (R1) and R1 sends the data on another router (R2) and a direct path from the host to R2 is available (that is, the host and R2 are on the same Ethernet segment), then R1 will send a redirect message to inform the host that the best route for the destination is via R2. The host should then send packets for the destination directly to R2. The router will still send the original datagram to the intended destination. However, if the datagram contains routing information, this message will not be sent even if a better route is available. RFC1122 states that redirects should only be sent by gateways and should not be sent by Internet hosts.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type = 5								Code				Header Checksum																			
IP Address																															
IP Header + First 8 Bytes of Original Datagram's Data																															

Type must be set to 5. The **code**, which specifies the reason for the redirection, include the following:

Code	Description
0	Redirect for Network.
1	Redirect for Host.
2	Redirect for Type of Service and Network.
3	Redirect for Type of Service and Host.

The **IP address** is the 32-bit address of the gateway to which the redirection should be sent. The **additional data** is included to allow the host to match the reply with the request that caused the redirection reply.

References

- RFC 792 - INTERNET CONTROL MESSAGE PROTOCOL
- RFC 1122 - Requirements for Internet Hosts—Communication Layers
- RFC 1812 - Requirements for IP Version 4 Routers

Explanation of ICMP Redirect Behavior [1]

References

[1] http://support.microsoft.com/kb/195686

Internet Control Message Protocol

The **Internet Control Message Protocol** (**ICMP**) is one of the core protocols of the Internet Protocol Suite. It is chiefly used by the operating systems of networked computers to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. ICMP can also be used to relay query messages.[1] It is assigned protocol number 1.[2]

ICMP[3] differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and traceroute).

ICMP for Internet Protocol version 4 (IPv4) is also known as ICMPv4. IPv6 has a similar protocol, ICMPv6.

Technical details

Internet Control Message Protocol is part of the Internet Protocol Suite as defined in RFC 792. ICMP messages are typically generated in response to errors in IP datagrams (as specified in RFC 1122) or for diagnostic or routing purposes. ICMP errors are always reported to the original source IP address of the originating datagram.[1]

An example ICMP error message is the Time To Live Exceeded message. Every machine (such as an intermediate router) that forwards an IP datagram has to decrement the time to live (TTL) field of the IP header by one. If the TTL reaches 0, an ICMP Time to live exceeded in transit message is sent to the source of the datagram.

Each ICMP message is encapsulated directly within a single IP datagram, and thus, like UDP, ICMP is unreliable.

Although ICMP messages are contained within standard IP datagrams, ICMP messages are usually processed as a special case, distinguished from normal IP processing, rather than processed as a normal sub-protocol of IP. In many cases, it is necessary to inspect the contents of the ICMP message and deliver the appropriate error message to the application that generated the original IP packet, the one that prompted the sending of the ICMP message.

Many commonly-used network utilities are based on ICMP messages. The tracert (traceroute), Pathping commands are implemented by transmitting UDP datagrams with specially set IP TTL header fields, and looking for ICMP Time to live exceeded in transit (above) and "Destination unreachable" messages generated in response. The related ping utility is implemented using the ICMP "Echo request" and "Echo reply" messages.

ICMP segment structure

Header

The ICMP header starts after the IPv4 header. All ICMP packets will have an 8-byte header and variable-sized data section. The first 4 bytes of the header will be consistent. The first byte is for the ICMP type. The second byte is for the ICMP code. The third and fourth bytes are a checksum of the entire ICMP message. The contents of the remaining 4 bytes of the header will vary based on the ICMP type and code.[1]

ICMP error messages contain a data section that includes the entire IP header plus the first 8 bytes of data from the IP datagram that caused the error message. The ICMP datagram is then encapsulated in a new IP datagram.[1]

Bits	0–7	8–15	16–23	24–31
0	Type	Code	Checksum	
32	Rest of Header			

- **Type –** ICMP type as specified below.
- **Code –** Subtype to the given type.
- **Checksum –** Error checking data. Calculated from the ICMP header+data, with value 0 for this field. The checksum algorithm is specified in RFC 1071 [4].
- **Rest of Header –** Four byte field. Will vary based on the ICMP type and code.

Padding data

Padding data follows the ICMP header (in octets):

- Windows "ping.exe" adds, by default, 32 bytes of padding
- The Linux "ping" utility adds, by default, 56 bytes of padding

List of permitted control messages (incomplete list)

Type	Code	Description
0 – Echo Reply[5]	0	Echo reply (used to ping)
1 and 2		*Reserved*
3 – Destination Unreachable[6]	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation required, and DF flag set
	5	Source route failed
	6	Destination network unknown
	7	Destination host unknown
	8	Source host isolated
	9	Network administratively prohibited
	10	Host administratively prohibited
	11	Network unreachable for TOS
	12	Host unreachable for TOS
	13	Communication administratively prohibited

4 – Source Quench	0	Source quench (congestion control)
5 – Redirect Message	0	Redirect Datagram for the Network
	1	Redirect Datagram for the Host
	2	Redirect Datagram for the TOS & network
	3	Redirect Datagram for the TOS & host
6		Alternate Host Address
7		*Reserved*
8 – Echo Request	0	Echo request (used to ping)
9 – Router Advertisement	0	Router Advertisement
10 – Router Solicitation	0	Router discovery/selection/solicitation
11 – Time Exceeded[7]	0	TTL expired in transit
	1	Fragment reassembly time exceeded
12 – Parameter Problem: Bad IP header	0	Pointer indicates the error
	1	Missing a required option
	2	Bad length
13 – Timestamp	0	Timestamp
14 – Timestamp Reply	0	Timestamp reply
15 – Information Request	0	Information Request
16 – Information Reply	0	Information Reply
17 – Address Mask Request	0	Address Mask Request
18 – Address Mask Reply	0	Address Mask Reply
19		*Reserved* for security
20 through 29		*Reserved* for robustness experiment
30 – Traceroute	0	Information Request
31		Datagram Conversion Error
32		Mobile Host Redirect
33		Where-Are-You (originally meant for IPv6)
34		Here-I-Am (originally meant for IPv6)
35		Mobile Registration Request
36		Mobile Registration Reply
37		Domain Name Request
38		Domain Name Reply
39		SKIP Algorithm Discovery Protocol, Simple Key-Management for Internet Protocol
40		Photuris, Security failures
41		ICMP for experimental mobility protocols such as Seamoby [RFC4065]
42 through 255		*Reserved*

(Sources: IANA ICMP Parameters [8] [9] and *Computer Networking – A Top-Down Approach* by Kurose and Ross)
//

References

[1] Forouzan, Behrouz A. (2007). *Data Communications And Networking* (Fourth ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 0-07-296775-7.
[2] "Protocol Numbers" (http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml). . Internet Assigned Numbers Authority. . Retrieved 2011-06-23.
[3] Postel, J. (September 1981). *Internet Control Message Protocol* (http://tools.ietf.org/html/rfc792). IETF. RFC 792. .
[4] http://tools.ietf.org/html/rfc1071
[5] http://tools.ietf.org/html/rfc792#page-14
[6] http://tools.ietf.org/html/rfc792#page-4
[7] http://tools.ietf.org/html/rfc792#page-6
[8] http://www.iana.org/assignments/icmp-parameters
[9] http://freebie.fatpipe.org/~mjb/Drawings/UDP_ICMP_Headers.png

External links

- RFCs
  - RFC 792, *Internet Control Message Protocol*
  - RFC 1122, *Requirements for Internet Hosts – Communication Layers*
  - RFC 1716, *Towards Requirements for IP Router*
- IANA (http://www.iana.org/assignments/icmp-parameters)
- ICMP Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/Icmp.pdf)
- ICMP ping simulation (http://www.visualland.net/view.php?cid=1124&protocol=ICMP&title=1.Ping basics)
- ICMP traceroute simulation (http://www.visualland.net/view.php?cid=1127&protocol=ICMP&title=6.Traceroute&ctype=1)
- IANA protocol numbers (http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml)

Internet Group Management Protocol

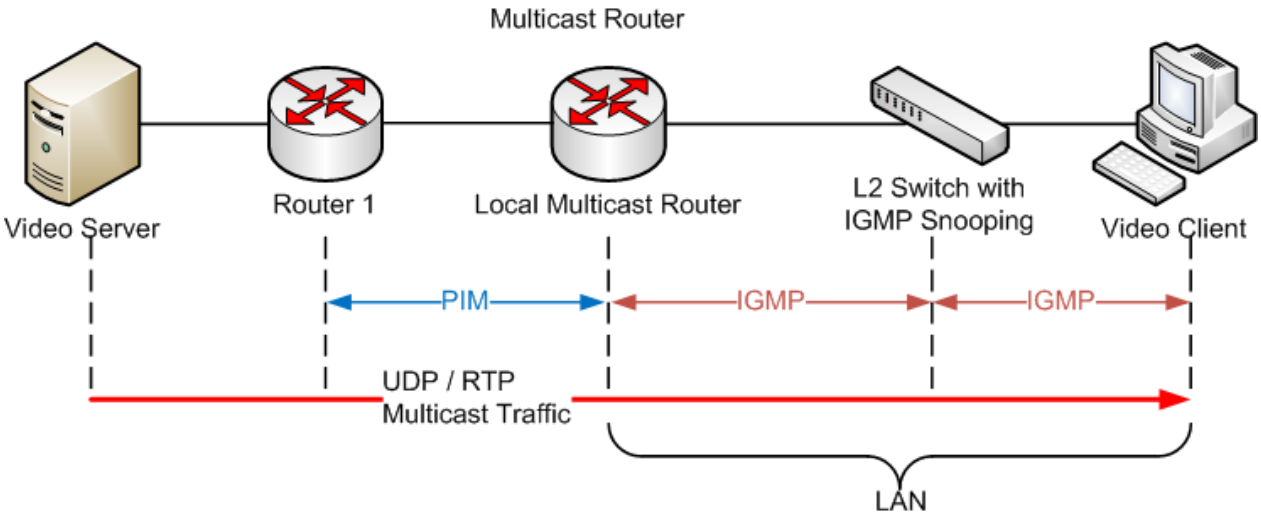
The **Internet Group Management Protocol** (**IGMP**) is a communications protocol used by hosts and adjacent routers on IP networks to establish multicast group memberships.

IGMP is an integral part of the IP multicast specification. It is analogous to ICMP for unicast connections. IGMP can be used for online streaming video and gaming, and allows more efficient use of resources when supporting these types of applications.

IGMP is used on IPv4 networks. Multicast management on IPv6 networks is handled by Multicast Listener Discovery (MLD) which uses ICMPv6 messaging contrary to IGMP's bare IP encapsulation.

Architecture

A network designed to deliver a multicast service using IGMP might use this basic architecture:



IGMP is used between the client computer and a local multicast router. Switches featuring IGMP snooping derive useful information by observing these IGMP transactions. Protocol Independent Multicast (PIM) is then used between the local and remote multicast routers, to direct multicast traffic from the multicast server to many multicast clients.

IGMP operates above the network layer, though it does not actually act as a transport protocol.[1]

Standards

There are three versions of IGMP, as defined by "Request for Comments" (RFC) documents of the Internet Engineering Task Force (IETF). IGMPv1 is defined by RFC 1112, IGMPv2 is defined by RFC 2236 and IGMPv3 was initially defined by RFC 3376 but has since been superseded by RFC 4604 which defines both IGMPv3 and MLDv2. IGMPv2 improves over IGMPv1 by adding the ability for a host to signal desire to leave a multicast group. IGMPv3 improves over IGMPv2 mainly by adding the ability to listen to multicast originating from a set of source IP addresses only.[2]

Host and router implementations

The IGMP protocol is implemented on a particular host and within a router. A host requests membership to a group through its local router while a router listens for these requests and periodically sends out subscription queries. The FreeBSD,[3] Linux[4] and Windows operating systems support IGMP at the host side.

For the server side implementation, the Linux case uses a daemon such as mrouted to act as an IGMP Linux router. There are also entire routing suites (such as XORP or Quagga), which turn an ordinary computer into a full-fledged multicast router.

Security

IGMP is vulnerable to some attacks,[5] [6] [7] [8] and firewalls commonly allow the user to disable it if not needed.

IGMPv3 packet structure

IGMP messages are carried in bare IP packets with IP protocol number 2.[9] There is no transport layer used with IGMP messaging, similar to ICMP for example.

Membership Query Message

Membership Queries are sent by multicast routers to determine which multicast addresses are of interest to systems attached to its network. Routers periodically send General Queries to refresh the group membership state for all systems on its network. Group-Specific Queries are used for determining the reception state for a particular multicast address. Group-and-Source-Specific Queries allow the router to determine if any systems desire reception of messages sent to a multicast group from a source address specified in a list of unicast addresses.

IGMPv3 packet structure

bit offset	0–3	4	5–7	8–15	16–31
0	Type = 0x11			Max Resp Code	Checksum
32	Group Address				
64	Resv	S	QRV	QQIC	Number of Sources (N)
96	Source Address [1]				
128	Source Address [2]				
	. . .				
	Source Address [N]				

Max Resp Code

This field specifies the maximum time (in 1/10 second) allowed before sending a responding report. If the number is below 128, the value is used directly. If the value is 128 or more, it is interpreted as an exponent with mantisse.

Checksum

This is the 16-bit one's complement of the one's complement sum of the entire IGMP message.

Group Address

This is the multicast address being queried when sending a Group-Specific or Group-and-Source-Specific Query. The field is zeroed when sending a General Query.

Resv

This field is reserved. It should be zeroed when sent and ignored when received.

S (Suppress Router-side Processing) Flag

When this flag is set, it indicates to receiving routers that they are to suppress the normal timer updates.

QRV (Querier's Robustness Variable)

If this is non-zero, it contains the Robustness Variable value used by the sender of the Query. Routers should update their Robustness Variable to match the most recently received Query unless the value is zero.

QQIC (Querier's Query Interval Code)

This code is used for specify the Query Interval value (in seconds) used by the querier. If the number is below 128, the value is used directly. If the value is 128 or more, it is interpreted as an exponent with mantisse.

Number of Sources (N)

This field specifies the number of source addresses present in the Query. For General and Group-Specific Queries, this value is zero. For Group-and-Source-Specific Queries, this value is non-zero, but limited by the network's MTU.

Source Address [i]

The Source Address [i] fields are a vector of n IP unicast addresses, where n is the value in the Number of Sources (N) field.

IGMPv2 packet structure

IGMPv2 packet structure

+	Bits 0–7	8–15	16–31
0	Type	Max Resp Time	Checksum
32	Group Address		

Defined by RFC 2236 [10]

Where:

- Type is Membership Query (0x11), Membership Report (IGMPv1: 0x12, IGMPv2: 0x16), Leave Group (0x17) IGMPv3 adds type Membership Report (0x22)
- Max Resp Time specifies the time limit for the corresponding report. The field has a resolution of 100 miliseconds, the value is taken directly. This field is meaningful only in Membership Query (0x11); in other messages it is set to 0 and ignored by the receiver.

Notes

[1] "IGMP, Internet Group Management Protocol" (http://www.networksorcery.com/enp/protocol/igmp.htm). Network Sorcery. . Retrieved 2010-11-18.
[2] "Internet Group Management Protocol Overview" (http://www.javvin.com/protocolIGMP.html). Javvin. . Retrieved 2010-11-18.
[3] IGMPv3 was added to FreeBSD in version 8.0.
[4] IGMPv3 was added in the Linux 2.5 kernel series.
[5] Spoofed IGMP report denial of service (http://www.securityfocus.com/bid/5020/info) vulnerability.
[6] Fragmented IGMP packet (http://support.microsoft.com/default.aspx?scid=kb;en-us;238329&sd=tech) may promote "Denial of Service" attack.
[7] IGMP Security Problem Statement and Requirements (http://www.securemulticast.org/GSEC/gsec3_ietf53_SecureIGMP1.pdf#search="igmp attacks").
[8] Microsoft Security Bulletin MS06-007: Vulnerability in TCP/IP Could Allow Denial of Service (913446) (http://www.microsoft.com/technet/security/Bulletin/MS06-007.mspx).
[9] RFC 3376 section 4

[10] http://tools.ietf.org/html/rfc2236#section-2

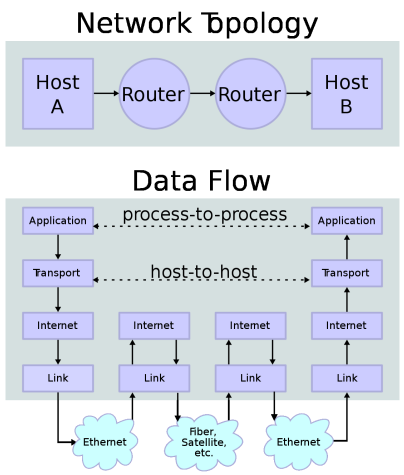
References

External links

- IPv4 Multicasting Tools and Settings on Microsoft TechNet (http://technet2.microsoft.com/WindowsServer/en/Library/fe09af2c-3deb-4c6c-a79f-35c6953a8c9d1033.mspx)
- Different version and details on IGMP (http://www.commsdesign.com/article/printableArticle.jhtml?articleID=52200253)

Internet Protocol

The **Internet Protocol** (**IP**) is the principal communications protocol used for relaying datagrams (packets) across an internetwork using the Internet Protocol Suite. Responsible for routing packets across network boundaries, it is the primary protocol that establishes the Internet.



Internet Protocol Suite in operation between two hosts connected via two routers and the corresponding layers used at each hop

IP is the primary protocol in the Internet Layer of the Internet Protocol Suite and has the task of delivering datagrams from the source host to the destination host solely based on their addresses. For this purpose, IP defines addressing methods and structures for datagram encapsulation.

Historically, IP was the connectionless datagram service in the original Transmission Control Program introduced by Vint Cerf and Bob Kahn in 1974, the other being the connection-oriented Transmission Control Protocol (TCP). The Internet Protocol Suite is therefore often referred to as TCP/IP.

The first major version of IP, now referred to as Internet Protocol Version 4 (IPv4) is the dominant protocol of the Internet, although the successor, Internet Protocol Version 6 (IPv6) is in active, growing deployment worldwide.

Function

The Internet Protocol is responsible for addressing hosts and routing datagrams (packets) from a source host to the destination host across one or more IP networks. For this purpose the Internet Protocol defines an addressing system that has two functions. Addresses identify hosts and provide a logical location service. Each packet is tagged with a header that contains the meta-data for the purpose of delivery. This process of tagging is also called encapsulation.

Reliability

The design principles of the Internet protocols assume that the network infrastructure is inherently unreliable at any single network element or transmission medium and that it is dynamic in terms of availability of links and nodes. No central monitoring or performance measurement facility exists that tracks or maintains the state of the network. For the benefit of reducing network complexity, the intelligence in the network is purposely mostly located in the end nodes of each data transmission, cf. end-to-end principle. Routers in the transmission path simply forward packets to the next known local gateway matching the routing prefix for the destination address.

As a consequence of this design, the Internet Protocol only provides best effort delivery and its service is characterized as *unreliable*. In network architectural language it is a *connection-less* protocol, in contrast to so-called connection-oriented modes of transmission. The lack of reliability permits various error conditions, such data corruption, packet loss and duplication, as well as out-of-order packet delivery. Since routing is dynamic for every packet and the network maintains no state of the path of prior packets, it is possible that some packets are routed on a longer path to their destination, resulting in improper sequencing at the receiver.

The only assistance that the Internet Protocol provides in Version 4 (IPv4) is to ensure that the IP packet header is error-free through computation of a checksum at the routing nodes. This has the side-effect of discarding packets with bad headers on the spot. In this case no notification is required to be sent to either end node, although a facility exists in the Internet Control Message Protocol (ICMP) to do so.

IPv6, on the other hand, has abandoned the use of IP header checksums for the benefit of rapid forwarding through routing elements in the network.

The resolution or correction of any of these reliability issues is the responsibility of an upper layer protocol. For example, to ensure in-order delivery the upper layer may have to cache data until it can be passed to the application.

In addition to issues of reliability, this dynamic nature and the diversity of the Internet and its components provide no guarantee that any particular path is actually capable of, or suitable for, performing the data transmission requested, even if the path is available and reliable. One of the technical constraints is the size of data packets allowed on a given link. An application must assure that it uses proper transmission characteristics. Some of this responsibility lies also in the upper layer protocols between application and IP. Facilities exist to examine the maximum transmission unit (MTU) size of the local link, as well as for the entire projected path to the destination when using IPv6. The IPv4 internetworking layer has the capability to automatically fragment the original datagram into smaller units for transmission. In this case, IP does provide re-ordering of fragments delivered out-of-order.[1]

Transmission Control Protocol (TCP) is an example of a protocol that will adjust its segment size to be smaller than the MTU. User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) disregard MTU size thereby forcing IP to fragment oversized datagrams.[2]

IP addressing and routing

Perhaps the most complex aspects of IP are IP addressing and routing. Addressing refers to how end hosts become assigned IP addresses and how subnetworks of IP host addresses are divided and grouped together. IP routing is performed by all hosts, but most importantly by internetwork routers, which typically use either interior gateway protocols (IGPs) or external gateway protocols (EGPs) to help make IP datagram forwarding decisions across IP connected networks.

IP routing is also common in local networks. For example, Ethernet switches sold today support IP multicast.[3] These switches primarily use IP addresses but also support MAC addresses for compatibility with older layer 2 Ethernet switches.

Version history

In May 1974, the Institute of Electrical and Electronic Engineers (IEEE) published a paper entitled "A Protocol for Packet Network Interconnection."[4] The paper's authors, Vint Cerf and Bob Kahn, described an internetworking protocol for sharing resources using packet-switching among the nodes. A central control component of this model was the "Transmission Control Program" (TCP) that incorporated both connection-oriented links and datagram services between hosts. The monolithic Transmission Control Program was later divided into a modular architecture consisting of the Transmission Control Protocol at the connection-oriented layer and the Internet Protocol at the internetworking (datagram) layer. The model became known informally as TCP/IP, although formally referenced as the Internet Protocol Suite.

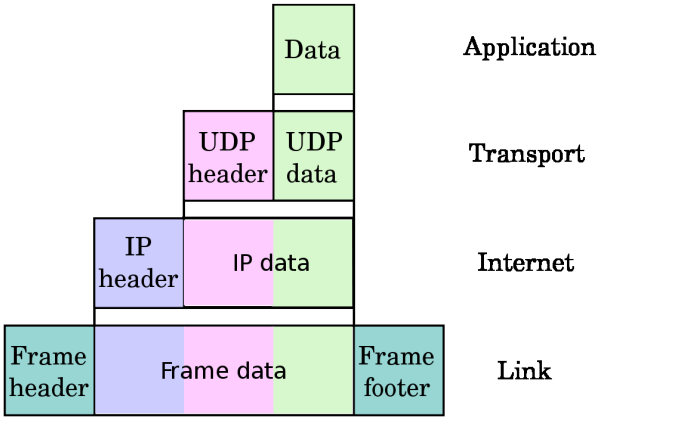
The Internet Protocol is one of the determining elements that define the Internet. The dominant internetworking protocol in the Internet Layer in use today is IPv4; with number 4 assigned as the formal protocol version number carried in every IP datagram. IPv4 is described in RFC 791 (1981).

The successor to IPv4 is IPv6. Its most prominent modification from version 4 is the addressing system. IPv4 uses 32-bit addresses (c. 4 billion, or 4.3×10^9, addresses) while IPv6 uses 128-bit addresses (c. 340 undecillion, or 3.4×10^{38} addresses). Although adoption of IPv6 has been slow, as of June 2008, all United States government systems have demonstrated basic infrastructure support for IPv6 (if only at the backbone level).[5]

Version numbers 0 through 3 were development versions of IPv4 used between 1977 and 1979. Version number 5 was used by the Internet Stream Protocol, an experimental streaming protocol. Version numbers 6 through 9 were proposed for various protocol models designed to replace IPv4: SIPP (Simple Internet Protocol Plus, known now as IPv6), TP/IX (RFC 1475), PIP (RFC 1621) and TUBA (TCP and UDP with Bigger Addresses, RFC 1347). Version number 6 was eventually chosen as the official assignment for the successor Internet protocol, subsequently standardized as IPv6.

A humorous Request for Comments that made an IPv9 protocol center of its storyline was published on April 1, 1994 by the IETF.[6] It was intended as an April Fool's Day joke. Other protocol proposals named "IPv9" and "IPv8" have also briefly surfaced, though these came with little or no support from the wider industry and academia.[7]

Reference diagrams



Sample encapsulation of application data from UDP to a Link protocol frame

Vulnerabilities

The Internet Protocol is vulnerable to a variety of attacks. A thorough vulnerability assessment, along with proposed mitigations, was published in 2008,[8] and is currently being pursued within the IETF.[9]

References

[1] Siyan, Karanjit. *Inside TCP/IP*, New Riders Publishing, 1997. ISBN 1-56205-714-6
[2] Basic Journey of a Packet (http://www.securityfocus.com/infocus/1870)
[3] Netgear ProSafe XSM7224S reference manual
[4] Vinton G. Cerf, Robert E. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. 22, No. 5, May 1974 pp. 637-648
[5] CIO council adds to IPv6 transition primer (http://www.gcn.com/print/25_16/41051-1.html), gcn.com
[6] RFC 1606: A Historical Perspective On The Usage Of IP Version 9. April 1, 1994.
[7] Theregister.com (http://www.theregister.co.uk/2004/07/06/ipv9_hype_dismissed/)
[8] Security Assessment of the Internet Protocol (IP) (http://www.cpni.gov.uk/Docs/InternetProtocol.pdf)

[9] Security Assessment of the Internet Protocol version 4 (IPv4) (http://tools.ietf.org/html/draft-ietf-opsec-ip-security)

External links

- Internet Protocol (http://www.dmoz.org/Computers/Internet/Protocols/) at the Open Directory Project
- RFC 791
- Data Communication Lectures of Manfred Lindner - Part IP Technology Basics (http://www.ict.tuwien.ac.at/lva/384.081/infobase/L30-IP_Technology_Basics_v4-6.pdf)
- Data Communication Lectures of Manfred Lindner - Part IP Technology Details (http://www.ict.tuwien.ac.at/lva/384.081/infobase/L31-IP_Technology_Details_v4-7.pdf)
- Data Communication Lectures of Manfred Lindner - Part IPv6 (http://www.ict.tuwien.ac.at/lva/384.081/infobase/L80-IPv6_v4-6.pdf)
- IPv6.com - Knowledge Center for Next Generation Internet IPv6 (http://www.ipv6.com)

IP address

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication.[1] An IP address serves two principal functions: host or network interface identification and location addressing. Its role has been characterized as follows: "*A name indicates what we seek. An address indicates where it is. A route indicates how to get there.*"[2]

The designers of the Internet Protocol defined an IP address as a 32-bit number[1] and this system, known as Internet Protocol Version 4 (IPv4), is still in use today. However, due to the enormous growth of the Internet and the predicted depletion of available addresses, a new addressing system (IPv6), using 128 bits for the address, was developed in 1995,[3] standardized as RFC 2460 in 1998,[4] and is being deployed worldwide since the mid-2000s.

IP addresses are binary numbers, but they are usually stored in text files and displayed in human-readable notations, such as 172.16.254.1 (for IPv4), and 2001:db8:0:1234:0:567:8:1 (for IPv6).

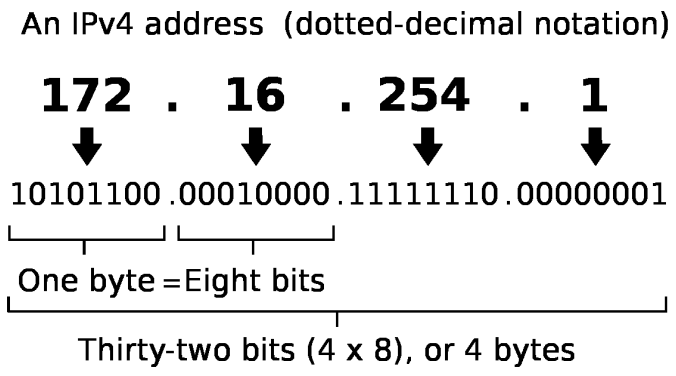
The Internet Assigned Numbers Authority (IANA) manages the IP address space allocations globally and delegates five regional Internet registries (RIRs) to allocate IP address blocks to local Internet registries (Internet service providers) and other entities.

IP versions

Two versions of the Internet Protocol (IP) are in use: IP Version 4 and IP Version 6. Each version defines an IP address differently. Because of its prevalence, the generic term *IP address* typically still refers to the addresses defined by IPv4. The gap in version sequence between IPv4 and IPv6 resulted from the assignment of number 5 to the experimental Internet Stream Protocol in 1979, which however was never referred to as IPv5.

IPv4 addresses

In IPv4 an address consists of 32 bits which limits the address space to 4294967296 (2^{32}) possible unique addresses. IPv4 reserves some addresses for special purposes such as private networks (~18 million addresses) or multicast addresses (~270 million addresses).



Decomposition of an IPv4 address from dot-decimal notation to its binary value.

IPv4 addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.16.254.1. Each part represents a group of 8 bits (octet) of the address. In some cases of technical writing, IPv4 addresses may be presented in various hexadecimal, octal, or binary representations.

IPv4 subnetting

In the early stages of development of the Internet Protocol,[1] network administrators interpreted an IP address in two parts: network number portion and host number portion. The highest order octet (most significant eight bits) in an address was designated as the *network number* and the remaining bits were called the *rest field* or *host identifier* and were used for host numbering within a network.

This early method soon proved inadequate as additional networks developed that were independent of the existing networks already designated by a network number. In 1981, the Internet addressing specification was revised with the introduction of classful network architecture.[2]

Classful network design allowed for a larger number of individual network assignments and fine-grained subnetwork design. The first three bits of the most significant octet of an IP address were defined as the *class* of the address. Three classes (*A*, *B*, and *C*) were defined for universal unicast addressing. Depending on the class derived, the network identification was based on octet boundary segments of the entire address. Each class used successively additional octets in the network identifier, thus reducing the possible number of hosts in the higher order classes (*B* and *C*). The following table gives an overview of this now obsolete system.

Class	Leading address bits (binary)	Range of first octet (decimal)	Network ID format	Host ID format	Number of networks	Number of addresses per network
A	000	0 - 127	a	b.c.d	$2^7 = 128$	$2^{24} = 16777216$
B	100 - 101	128 - 191	a.b	c.d	$2^{14} = 16384$	$2^{16} = 65536$
C	110	192 - 223	a.b.c	d	$2^{21} = 2097152$	$2^8 = 256$

|+ Historical classful network architecture

Classful network design served its purpose in the startup stage of the Internet, but it lacked scalability in the face of the rapid expansion of the network in the 1990s. The class system of the address space was replaced with Classless Inter-Domain Routing (CIDR) in 1993. CIDR is based on variable-length subnet masking (VLSM) to allow allocation and routing based on arbitrary-length prefixes.

Today, remnants of classful network concepts function only in a limited scope as the default configuration parameters of some network software and hardware components (e.g. netmask), and in the technical jargon used in network administrators' discussions.

IPv4 private addresses

Early network design, when global end-to-end connectivity was envisioned for communications with all Internet hosts, intended that IP addresses be uniquely assigned to a particular computer or device. However, it was found that this was not always necessary as private networks developed and public address space needed to be conserved.

Computers not connected to the Internet, such as factory machines that communicate only with each other via TCP/IP, need not have globally unique IP addresses. Three ranges of IPv4 addresses for private networks were reserved in RFC 1918. These addresses are not routed on the Internet and thus their use need not be coordinated with an IP address registry.

Today, when needed, such private networks typically connect to the Internet through network address translation (NAT).

IANA-reserved private IPv4 network ranges

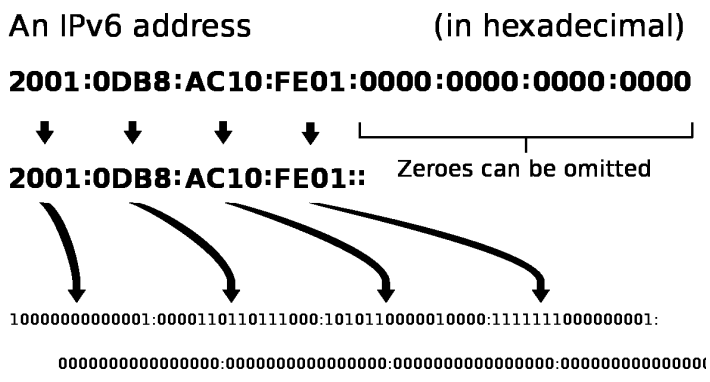
	Start	End	No. of addresses
24-bit Block (/8 prefix, 1 × A)	10.0.0.0	10.255.255.255	16777216
20-bit Block (/12 prefix, 16 × B)	172.16.0.0	172.31.255.255	1048576
16-bit Block (/16 prefix, 256 × C)	192.168.0.0	192.168.255.255	65536

Any user may use any of the reserved blocks. Typically, a network administrator will divide a block into subnets; for example, many home routers automatically use a default address range of 192.168.0.0 - 192.168.0.255 (192.168.0.0/24).

IPv4 address exhaustion

IPv4 address exhaustion is the decreasing supply of unallocated Internet Protocol Version 4 (IPv4) addresses available at the Internet Assigned Numbers Authority (IANA) and the regional Internet registries (RIRs) for assignment to end users and local Internet registries, such as Internet service providers. IANA's primary address pool was exhausted on February 3, 2011 when the last 5 blocks were allocated to the 5 RIRs.[5] [6] APNIC was the first RIR to exhaust its regional pool on 15 April 2011, except for a small amount of address space reserved for the transition to IPv6, intended be allocated in a restricted process[7]

IPv6 addresses



Decomposition of an IPv6 address from hexadecimal representation to its binary value.

The rapid exhaustion of IPv4 address space, despite conservation techniques, prompted the Internet Engineering Task Force (IETF) to explore new technologies to expand the Internet's addressing capability. The permanent solution was deemed to be a redesign of the Internet Protocol itself. This next generation of the Internet Protocol, intended to replace IPv4 on the Internet, was eventually named *Internet Protocol Version 6* (IPv6) in 1995[3] [4] The address size was increased from 32 to 128 bits or 16 octets. This, even with a generous assignment of network blocks, is deemed sufficient for the foreseeable future. Mathematically, the new address space provides the potential for a maximum of 2^{128}, or about 3.403×10^{38} unique addresses.

The new design is not intended to provide a sufficient quantity of addresses on its own, but rather to allow efficient aggregation of subnet routing prefixes to occur at routing nodes. As a result, routing table sizes are smaller, and the smallest possible individual allocation is a subnet for 2^{64} hosts, which is the square of the size of the entire IPv4 Internet. At these levels, actual address utilization rates will be small on any IPv6 network segment. The new design also provides the opportunity to separate the addressing infrastructure of a network segment — that is the local administration of the segment's available space — from the addressing prefix used to route external traffic for a network. IPv6 has facilities that automatically change the routing prefix of entire networks, should the global connectivity or the routing policy change, without requiring internal redesign or renumbering.

The large number of IPv6 addresses allows large blocks to be assigned for specific purposes and, where appropriate, to be aggregated for efficient routing. With a large address space, there is not the need to have complex address conservation methods as used in Classless Inter-Domain Routing (CIDR).

Many modern desktop and enterprise server operating systems include native support for the IPv6 protocol, but it is not yet widely deployed in other devices, such as home networking routers, voice over IP (VoIP) and multimedia equipment, and network peripherals.

IPv6 private addresses

Just as IPv4 reserves addresses for private or internal networks, blocks of addresses are set aside in IPv6 for private addresses. In IPv6, these are referred to as unique local addresses (ULA). RFC 4193 sets aside the routing prefix fc00::/7 for this block which is divided into two /8 blocks with different implied policies The addresses include a 40-bit pseudorandom number that minimizes the risk of address collisions if sites merge or packets are misrouted.[8]

Early designs used a different block for this purpose (fec0::), dubbed site-local addresses.[9] However, the definition of what constituted *sites* remained unclear and the poorly defined addressing policy created ambiguities for routing. This address range specification was abandoned and must not be used in new systems.[10]

Addresses starting with fe80:, called link-local addresses, are assigned to interfaces for communication on the link only. The addresses are automatically generated by the operating system for each network interface. This provides instant and automatic network connectivity for any IPv6 host and means that if several hosts connect to a common hub or switch, they have a communication path via their link-local IPv6 address. This feature is used in the lower layers of IPv6 network administration (e.g. Neighbor Discovery Protocol).

None of the private address prefixes may be routed on the public Internet.

IP subnetworks

IP networks may be divided into subnetworks in both IPv4 and IPv6. For this purpose, an IP address is logically recognized as consisting of two parts: the *network prefix* and the *host identifier*, or *interface identifier* (IPv6). The subnet mask or the CIDR prefix determines how the IP address is divided into network and host parts.

The term *subnet mask* is only used within IPv4. Both IP versions however use the Classless Inter-Domain Routing (CIDR) concept and notation. In this, the IP address is followed by a slash and the number (in decimal) of bits used for the network part, also called the *routing prefix*. For example, an IPv4 address and its subnet mask may be 192.0.2.1 and 255.255.255.0, respectively. The CIDR notation for the same IP address and subnet is 192.0.2.1/24, because the first 24 bits of the IP address indicate the network and subnet.

IP address assignment

Internet Protocol addresses are assigned to a host either anew at the time of booting, or permanently by fixed configuration of its hardware or software. Persistent configuration is also known as using a *static IP address*. In contrast, in situations when the computer's IP address is assigned newly each time, this is known as using a *dynamic IP address*.

Methods

Static IP addresses are manually assigned to a computer by an administrator. The exact procedure varies according to platform. This contrasts with dynamic IP addresses, which are assigned either by the computer interface or host software itself, as in Zeroconf, or assigned by a server using Dynamic Host Configuration Protocol (DHCP). Even though IP addresses assigned using DHCP may stay the same for long periods of time, they can generally change. In some cases, a network administrator may implement dynamically assigned static IP addresses. In this case, a DHCP server is used, but it is specifically configured to always assign the same IP address to a particular computer. This allows static IP addresses to be configured centrally, without having to specifically configure each computer on the network in a manual procedure.

In the absence or failure of static or stateful (DHCP) address configurations, an operating system may assign an IP address to a network interface using state-less auto-configuration methods, such as Zeroconf.

Uses of dynamic addressing

Dynamic IP addresses are most frequently assigned on LANs and broadband networks by Dynamic Host Configuration Protocol (DHCP) servers. They are used because it avoids the administrative burden of assigning specific static addresses to each device on a network. It also allows many devices to share limited address space on a network if only some of them will be online at a particular time. In most current desktop operating systems, dynamic IP configuration is enabled by default so that a user does not need to manually enter any settings to connect to a network with a DHCP server. DHCP is not the only technology used to assign dynamic IP addresses. Dialup and some broadband networks use dynamic address features of the Point-to-Point Protocol.

Sticky dynamic IP address

A *sticky dynamic IP address* is an informal term used by cable and DSL Internet access subscribers to describe a dynamically assigned IP address that seldom changes. The addresses are usually assigned with the DHCP protocol. Since the modems are usually powered-on for extended periods of time, the address leases are usually set to long periods and simply renewed upon expiration. If a modem is turned off and powered up again before the next expiration of the address lease, it will most likely receive the same IP address.

Address autoconfiguration

RFC 3330 defines an address block, 169.254.0.0/16, for the special use in link-local addressing for IPv4 networks. In IPv6, every interface, whether using static or dynamic address assignments, also receives a local-link address automatically in the block fe80::/10.

These addresses are only valid on the link, such as a local network segment or point-to-point connection, that a host is connected to. These addresses are not routable and like private addresses cannot be the source or destination of packets traversing the Internet.

When the link-local IPv4 address block was reserved, no standards existed for mechanisms of address autoconfiguration. Filling the void, Microsoft created an implementation that is called Automatic Private IP Addressing (APIPA). Due to Microsoft's market power, APIPA has been deployed on millions of machines and has, thus, become a de facto standard in the industry. Many years later, the IETF defined a formal standard for this functionality, RFC 3927, entitled *Dynamic Configuration of IPv4 Link-Local Addresses*.

Uses of static addressing

Some infrastructure situations have to use static addressing, such as when finding the Domain Name System (DNS) host that will translate domain names to IP addresses. Static addresses are also convenient, but not absolutely necessary, to locate servers inside an enterprise. An address obtained from a DNS server comes with a time to live, or caching time, after which it should be looked up to confirm that it has not changed. Even static IP addresses do change as a result of network administration (RFC 2072)

Public addresses

A *public IP address* in common parlance is synonymous with a, globally routable unicast IP address.

Both IPv4 and IPv6 define address ranges that are reserved for private networks and link-local addressing. The term public IP address often used exclude these types of addresses.

Modifications to IP addressing

IP blocking and firewalls

Firewalls perform Internet Protocol blocking to protect networks from unauthorized access. They are common on today's Internet. They control access to networks based on the IP address of a client computer. Whether using a blacklist or a whitelist, the IP address that is blocked is the perceived IP address of the client, meaning that if the client is using a proxy server or network address translation, blocking one IP address may block many individual computers.

IP address translation

Multiple client devices can appear to share IP addresses: either because they are part of a shared hosting web server environment or because an IPv4 network address translator (NAT) or proxy server acts as an intermediary agent on behalf of its customers, in which case the real originating IP addresses might be hidden from the server receiving a request. A common practice is to have a NAT hide a large number of IP addresses in a private network. Only the "outside" interface(s) of the NAT need to have Internet-routable addresses.[11]

Most commonly, the NAT device maps TCP or UDP port numbers on the outside to individual private addresses on the inside. Just as a telephone number may have site-specific extensions, the port numbers are site-specific extensions to an IP address.

In small home networks, NAT functions usually take place in a residential gateway device, typically one marketed as a "router". In this scenario, the computers connected to the router would have 'private' IP addresses and the router would have a 'public' address to communicate with the Internet. This type of router allows several computers to share one public IP address.

Diagnostic tools

Computer operating systems provide various diagnostic tools to examine their network interface and address configuration. Windows provides the command-line interface tools `ipconfig` and `netsh` and users of Unix-like systems can use `ifconfig`, `netstat`, `route`, `lanstat`, `ifstat`, or `iproute2` utilities to accomplish the task.

References

[1] RFC 760, *DOD Standard Internet Protocol* (January 1980)
[2] RFC 791, *Internet Protocol - DARPA Internet Program Protocol Specification* (September 1981)
[3] RFC 1883, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden (December 1995)
[4] RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden, The Internet Society (December 1998)
[5] Smith, Lucie; Lipner, Ian (3 February 2011). "Free Pool of IPv4 Address Space Depleted" (http://www.nro.net/news/ipv4-free-pool-depleted). Number Resource Organization. . Retrieved 3 February 2011.
[6] ICANN,nanog mailing list. "Five /8s allocated to RIRs - no unallocated IPv4 unicast /8s remain" (http://mailman.nanog.org/pipermail/nanog/2011-February/032107.html). .
[7] Asia-Pacific Network Information Centre (15 April 2011). "APNIC IPv4 Address Pool Reaches Final /8" (http://www.apnic.net/publications/news/2011/final-8). . Retrieved 15 April 2011.
[8] RFC 4193 section 3.2.1
[9] RFC 3513
[10] RFC 3879
[11] Comer, Douglas (2000). *Internetworking with TCP/IP:Principles, Protocols, and Architectures -- 4th ed.* (http://www.cs.purdue.edu/homes/dec/netbooks.html). Upper Saddle River, NJ: Prentice Hall. p. 394. ISBN 0-13-018380-6. .

External links

- IP (http://www.dmoz.org/Computers/Internet/Protocols/IP/) at the Open Directory Project
- "Understanding IP Addressing: Everything You Ever Wanted To Know" (http://web.archive.org/web/20100821112028/http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf). Archived from the original (http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf) on 2010-08-21.

IP forwarding algorithm

The IP forwarding algorithm is related to the topic of routing and gives a more directed approach in forwarding datagrams over a network. In order to achieve a successful transfer of data the algorithm uses a routing table to select a next-hop router as the next destination for a datagram. The IP address that is selected is known as the *next-hop address.*[1]

The IP Forwarding Algorithm states:

Given a destination IP address, D, and network prefix, N:

```
if (N matches a directly connected network address)

    Deliver datagram to D over that network;

else if (N does not match a network address
         and routing table contains route for N)

    Send datagram to next-hop address listed in the routing table;

else if (N does not match a network
         and routing table does not contain route for N
         and there exists a default route)

    Send datagram to default route;

else

    Send forwarding error message;
```

References

[1] Internetworking with TCP/IP: Principles, protocols, and architecture By Douglas Comer

IP fragmentation

The Internet Protocol (**IP**) implements datagram **fragmentation**, so that packets may be formed that can pass through a link with a smaller maximum transmission unit (MTU) than the original datagram size.

RFC 791 describes the procedure for IP fragmentation, and transmission and reassembly of datagrams.[1] RFC 815 describes a simplified reassembly algorithm.[2]

The *Identification* field, and *Fragment* offset field along with *Don't Fragment* and *More Fragment* flags in the IP protocol header are used for fragmentation and reassembly of IP datagrams.

In a case where a router receives a protocol data unit (PDU) larger than the next hop's MTU, it has two options if the transport is IPv4. Drop the PDU and send an Internet Control Message Protocol (ICMP) message which indicates the condition *Packet too Big*, or to fragment the IP packet and send it over the link with a smaller MTU. IPv6 hosts are required to determine the optimal Path MTU before sending packets.

If a receiving host receives a fragmented IP packet, it has to reassemble the datagram and pass it to the higher protocol layer. Reassembly is intended to happen in the receiving host but in practice it may be done by an intermediate router, for example, network address translation *may* need to re-assemble fragments in order to translate data streams, e.g. the FTP control protocol, as described in RFC 2993.[3]

IP fragmentation can cause excessive retransmissions when fragments encounter packet loss and reliable protocols such as TCP must retransmit all of the fragments in order to recover from the loss of a single fragment.[4] Thus, senders typically use two approaches to decide the size of IP datagrams to send over the network. The first is for the sending host to send an IP datagram of size equal to the MTU of the first hop of the source destination pair. The second is to run the path MTU discovery algorithm,[5] described in RFC 1191, to determine the path MTU between two IP hosts, so that IP fragmentation can be avoided.

IPv4 and IPv6 differences

The details of the fragmentation mechanism, as well as the overall architectural approach to fragmentation, are different between IPv4, the first official version of the Internet Protocol, and IPv6, the newer version. In IPv4, routers perform fragmentation, whereas in IPv6, routers do not fragment, but drop the packets that are larger than the MTU. Though the header formats are different for IPv4 and IPv6, analogous fields are used for fragmentation, so the algorithm can be reused for fragmentation and reassembly.

In IPv4, hosts must make a best-effort attempt to reassemble fragmented IP datagrams with a total reassembled size of up to 576 bytes - equal to the minimum MTU for IPv4. They may also attempt to reassemble fragmented IP datagrams larger than 576 bytes, but they are also permitted to silently discard such larger datagrams. In IPv6, this minimum capability is increased to 1280 bytes [6] - larger than the minimum MTU for IPv4.

References

[1] RFC 791, *Internet Protocol*, Information Sciences Institute (September 1981)
[2] RFC 815, *IP Datagram Reassembly Algorithms*, David D. Clark (July 1982)
[3] RFC 2993, *Architectural Implications of NAT* (November 2000)
[4] Christopher A. Kent, Jeffrey C. Mogul. "Fragmentation Considered Harmful" (http://www.cs.binghamton.edu/~nael/classes/cs528/fragment.pdf). .
[5] RFC 1191, *Path MTU Discovery* (November 1990)
[6] RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden (December 1998)

External links

- IP Fragmentation (http://penguin.dcs.bbk.ac.uk/academic/networks/network-layer/fragmentation/index.php)
- What is packet fragmentation? (http://www.tech-faq.com/packet-fragmentation.shtml)

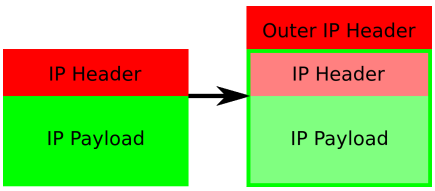
IP in IP

IP in IP is an IP tunneling protocol that encapsulates one IP packet in another IP packet. To encapsulate an IP packet in another IP packet, an outer header is added with SourceIP, the entry point of the tunnel and the Destination point, the exit point of the tunnel. While doing this the inner packet is not modified except the TTL field which is decremented. The Don't Fragment and the Type Of Service field should be copied to the outer packet. If the packet size is greater than the Path MTU, the packet is fragmented in the encapsulator as the outer header should be included. The decapsulator will reassemble the packet.

IP Packet encapsulated in IP Packet

Outer IP header	Inner IP header	IP Payload

Outer IP header have the following fields:



IP in IP Encapsulation

Version: 4 bits
This field is the Protocol version number. It is always 4 as IP in IP is supported for IPv4

Header Length: 4 bits
This field is the length of outer IP header

Type of Service (TOS): 8 bits
This field is copied from the inner IP header

Total Length: 16 bits
This field is the length of the encapsulated IP header (including Outer IP header, Inner IP header, IP Payload)

Identification: 16 bits
This field is used to identify the fragments of a datagram which will be helpful while reassembling the datagrams as encapsulator might fragment the datagram. For the outer IP header a new number is generated.

Flags: 3 bits

R	DF	MF

R: 1 bit
This bit is reserved and should be 0.
DF: 1 bit
This field specifies whether the datagram can be fragmented or not. If this bit is set to 1 in the inner header, then the outer header also have this bit set to 1 saying that this datagram cannot be fragmented. If this bit is set to 0 in the inner header, then the outer header may set to 0/1.
MF: 1 bit
This field is required when the datagram is fragmented saying whether the datagram contains some more fragments. This field is not copied from inner header.

Fragment Offset: 13 bits
This field is used while reassembling the fragments.

Time To Live (TTL): 8 bits
This field is used to track the lifetime of the datagram. The inner header TTL is decremented before encapsulation and is not changed in decapsulator. The outer header TTL is set to value such that the datagram is delivered to tunnel end point.

Protocol: 8 bits
This field indicates the next protocol following this datagram. The value is set to 4. Most cases the protocol will be IPv4 unless there is some additional headers for encapsulated packet.

Header Checksum: 16 bits
This field is the IP checksum of outer header.

Source IP Address: 32 bits
This field is the IP address of the encapsulator

Destination IP Address: 32 bits
This field is the IP address of the decapsulator

Options: Variable length
This field in general is NOT copied from the inner IP header. New options can be added.

Padding. Variable length.
This field is used to fill the datagram so that IP Payload starts on a 32 bit boundary.

Handling ICMP Messages

After sending a datagram, there may be chances of encapsulator getting ICMP messages from intermediate nodes. The encapsulator takes action for ICMP message depending on the Type and Code of the ICMP messages. The following are the ICMP messages with Type and Code as well as the action taken by the encapsulator:

- **Destination Unreachable (Type 3)**
  - **Network Unreachable (Code 0)** - ICMP error message with Type 3 Code 0 should be sent to original sender.
  - **Host Unreachable (Code 1)** - ICMP error message with Type 3 Code 0 should be sent to original sender.
  - **Protocol Unreachable (Code 2)** - ICMP error message with Type 3 Code 0 or 1 should be sent to original sender.
  - **Port Unreachable (Code 3)** - No ICMP error message should be sent to original sender.
  - **Datagram too big (Code 4)** - ICMP error message with Type 3 Code 4 should be sent to original sender.
  - **Source Route Failed (Code 5)** - No ICMP error message should be sent to original sender.
- **Source Quence (Type 4)** - No ICMP error message should be sent to original sender
- **Redirect (Type 5)** - No ICMP error message should be sent to original sender
- **Time Exceeded (Type 6)** - ICMP error message with Type 3 Code 1 should be sent to original sender (Case of Routing loops within tunnels)

- **Parameter Problem (Type 12)**
  - If the error message points to a field copied from unencapsulated datagram, then the ICMP error message should be sent to original sender.
  - If the error message points to a field inserted by encapsulator, then the ICMP error message should not be sent to original sender

Handling Tunnel Loops

Loops may occurs within a tunnel due to the following reasons:

1. when the source IP of the datagram is same as the router's IP address in any of the interface.
2. when the source IP of the datagram is same as the tunnel end point.

In both the cases, the router MUST NOT tunnel datagram. Instead it should discard the datagram.

Tunnel Management

For ICMP messages intermediate routers return 64 bits of the datagram beyond the IP header which is not enough to copy Inner header. So the encapsulator will not be able to relay the appropriate message to the original sender. But this can be handled by maintaining the soft state of the tunnel. The soft state information maintained by the tunnel includes the following:

- MTU of the tunnel
- TTL of the tunnel
- Reachability of the decapsulator

References

RFC 2003

IPv4

Internet Protocol version 4 (**IPv4**) is the fourth revision in the development of the Internet Protocol (IP) and the first version of the protocol to be widely deployed. Together with IPv6, it is at the core of standards-based internetworking methods of the Internet. IPv4 is still by far the most widely deployed Internet Layer protocol (As of 2011, IPv6 deployment is still in its infancy).

IPv4 is described in IETF publication RFC 791 (September 1981), replacing an earlier definition (RFC 760, January 1980).

IPv4 is a connectionless protocol for use on packet-switched Link Layer networks (e.g., Ethernet). It operates on a best effort delivery model, in that it does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol , such as the Transmission Control Protocol (TCP).

Addressing

IPv4 uses 32-bit (four-byte) addresses, which limits the address space to 4294967296 (2^{32}) addresses. However, some address blocks are reserved for special purposes such as private networks (~18 million addresses) and multicast addresses (~270 million addresses). This reduces the number of addresses that may be allocated for routing on the public Internet. As addresses are assigned to end users, an IPv4 address shortage has been developing. Network addressing changes by classful network design, Classless Inter-Domain Routing, and network address translation (NAT) have contributed to delay significantly the inevitable exhaustion which occurred on February 3, 2011 when IANA allocated the last five blocks to the five regional Internet registries (RIRs).

This limitation stimulated the development of IPv6 in the 1990s, which has been in commercial deployment since 2006.

Address representations

IPv4 addresses may be written in any notation expressing a 32-bit integer value, but for human convenience, they are most often written in dot-decimal notation, which consists of four octets of the address expressed individually in decimal and separated by periods.

The following table shows several representation formats:

Notation	Value	Conversion from dot-decimal
Dot-decimal notation	`192.0.2.235`	N/A
Dotted Hexadecimal[1]	`0xC0.0x00.0x02.0xEB`	Each octet is individually converted to hexadecimal form
Dotted Octal[1]	`0300.0000.0002.0353`	Each octet is individually converted into octal
Hexadecimal	`0xC00002EB`	Concatenation of the octets from the dotted hexadecimal
Decimal	`3221226219`	The 32-bit number expressed in decimal
Octal	`030000001353`	The 32-bit number expressed in octal

Additionally, in dotted format, each octet can be of any of the different bases. For example, `192.0x00.0002.235` is a valid (though unconventional) equivalent to the above addresses.

Allocation

Originally, an IP address was divided into two parts, the network identifier represented in the most significant (highest order) octet of the address and the host identifier using the rest of the address. The latter was therefore also called the *rest field*. This enabled the creation of a maximum of 256 networks. This was quickly found to be inadequate.

To overcome this limit, the high order octet of the addresses was redefined to create a set of *classes* of networks, in a system which later became known as classful networking. The system defined five classes, Class A, B, C, D, and E. The Classes A, B, and C had different bit lengths for the new network identification. The rest of an address was used as previously to identify a host within a network, which meant that each network class had a different capacity to address hosts. Class D was allocated for multicast addressing and Class E was reserved for future applications.

Starting around 1985, methods were devised to allow IP networks to be subdivided. The concept of the *variable-length subnet mask* (VLSM) was introduced which allowed flexible subdivision into varying network sizes.[2] [3]

Around 1993, this system of classes was officially replaced with Classless Inter-Domain Routing (CIDR), and the class-based scheme was dubbed *classful*, by contrast.

CIDR was designed to permit repartitioning of any address space so that smaller or larger blocks of addresses could be allocated to users. The hierarchical structure created by CIDR is managed by the Internet Assigned Numbers Authority (IANA) and the regional Internet registries (RIRs). Each RIR maintains a publicly-searchable WHOIS database that provides information about IP address assignments.

Special-use addresses

Reserved address blocks

CIDR address block	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 1700
10.0.0.0/8	Private network	RFC 1918
127.0.0.0/8	Loopback	RFC 5735
169.254.0.0/16	Link-Local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	Reserved (IANA)	RFC 5735
192.0.2.0/24	TEST-NET-1, Documentation and example code	RFC 5735
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, Documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, Documentation and examples	RFC 5737
224.0.0.0/4	Multicasts (former Class D network)	RFC 3171
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

Private networks

Of the approximately four billion addresses allowed in IPv4, three ranges of address are reserved for use in private networks. These ranges are not routable outside of private networks and private machines cannot directly communicate with public networks. They can, however, do so through network address translation.

The following are the three ranges reserved for private networks (RFC 1918):

Name	Address range	Number of addresses	*Classful* description	Largest CIDR block
24-bit block	10.0.0.0–10.255.255.255	16777216	Single Class A	10.0.0.0/8
20-bit block	172.16.0.0–172.31.255.255	1048576	Contiguous range of 16 Class B blocks	172.16.0.0/12
16-bit block	192.168.0.0–192.168.255.255	65536	Contiguous range of 256 Class C blocks	192.168.0.0/16

Virtual private networks

Packets with a private destination address are ignored by all public routers. Therefore, it is not possible to communicate directly between two private networks (e.g., two branch offices) via the public Internet. This requires the use of IP tunnels or a virtual private network (VPN).

VPNs establish tunneling connections across the public network such that the endpoints of the tunnel function as routers for packets from the private network. In this routing function the host encapsulates packets in a protocol layer with packet headers acceptable in the public network so that they may be delivered to the opposing tunnel end point where the additional protocol layer is removed and the packet is delivered locally to its intended destination.

Optionally, encapsulated packets may be encrypted to secure the data while it travels over the public network.

Link-local addressing

RFC 5735 defines an address block, 169.254.0.0/16, for the special use in link-local addressing. These addresses are only valid on the link, such as a local network segment or point-to-point connection, that a host is connected to. These addresses are not routable and like private addresses cannot be the source or destination of packets traversing the Internet. Link-local addresses are primarily used for address autoconfiguration (Zeroconf) when a host cannot obtain an IP address from a DHCP server or other internal configuration methods.

When the address block was reserved, no standards existed for mechanisms of address autoconfiguration. Filling the void, Microsoft created an implementation called Automatic Private IP Addressing (APIPA). Due to Microsoft's market power, APIPA has been deployed on millions of machines and has, thus, become a de facto standard in the industry. Many years later, the IETF defined a formal standard for this functionality, RFC 3927, entitled *Dynamic Configuration of IPv4 Link-Local Addresses*.

Localhost

The address range 127.0.0.0–127.255.255.255 (127.0.0.0/8 in CIDR notation) is reserved for localhost communication. Addresses within this range should never appear outside a host computer and packets sent to this address are returned as incoming packets on the same virtual network device (known as loopback).

Addresses ending in 0 or 255

Networks with subnet masks of at least 24 bits, i.e. Class C networks in classful networking, and networks with CIDR prefixes /24 to /32 (255.255.255.0–255.255.255.255) may not have an address ending in 0 or 255.

Classful addressing prescribed only three possible subnet masks: Class A, 255.0.0.0 or /8; Class B, 255.255.0.0 or /16; and Class C, 255.255.255.0 or /24. For example, in the subnet 192.168.5.0/255.255.255.0 (192.168.5.0/24) the identifier 192.168.5.0 commonly is used to refer to the entire subnet. To avoid ambiguity in representation, the address ending in the octet *0* is reserved.

A broadcast address is an address that allows information to be sent to all interfaces in a given subnet, rather than a specific machine. Generally, the broadcast address is found by obtaining the bit complement of the subnet mask and performing a bitwise OR operation with the network identifier. In other words, the broadcast address is the last address in the address range of the subnet. For example, the broadcast address for the network 192.168.5.0 is 192.168.5.255. For networks of size /24 or larger, the broadcast address always ends in 255.

However, this does not mean that every address ending in 0 or 255 cannot be used as a host address. For example, in the case of a /16 subnet 192.168.0.0/255.255.0.0, equivalent to the address range 192.168.0.0–192.168.255.255, the broadcast address is 192.168.255.255. However, one may assign 192.168.1.255, 192.168.2.255, etc. 192.168.0.0 is the network identifier which should not be assigned to an interface,[4] , but 192.168.1.0, 192.168.2.0, etc. may be assigned.

In the past, conflict between network addresses and broadcast addresses arose because some software used non-standard broadcast addresses with zeros instead of ones.[5]

In networks smaller than /24, broadcast addresses do not necessarily end with 255. For example, a CIDR subnet 203.0.113.16/28 has the broadcast address 203.0.113.31.

Address resolution

Hosts on the Internet are usually known by names, e.g., www.example.com, not primarily by their IP address, which is used for routing and network interface identification. The use of domain names requires translating, called *resolving*, them to addresses and vice versa.

The translation between addresses and domain names is performed by the Domain Name System (DNS), a hierarchical, distributed naming system which allows for subdelegation of name spaces to other DNS servers. DNS is often described in analogy to the telephone system directory information systems in which subscriber names are translated to telephone numbers.

Address space exhaustion

Since the 1980s it was apparent that the pool of available IPv4 addresses was depleted at a rate that was not initially anticipated in the original design of the network address system.[6] The apparent threat of exhaustion was the motivation for remedial technologies, such as the introduction of classful networks, the creation of Classless Inter-Domain Routing (CIDR) methods, and network address translation (NAT), and finally for the redesign of the Internet Protocol, based on a larger address format (IPv6).

Several market forces have driven the acceleration of IPv4 address exhaustion:

- Rapidly growing number of Internet users
- Always-on devices — ADSL modems, cable modems
- Mobile devices — laptop computers, PDAs, mobile phones

A variety of technologies introduced during the growth of the Internet have been applied to mitigate IPv4 address exhaustion and its effects, such as:

- Network address translation (NAT) is a technology that masquerades an entire, private network with a single public IP address, permitting the use of private addresses within the private network.
- Use of private networks
- Dynamic Host Configuration Protocol (DHCP)
- Name-based virtual hosting of web sites
- Tighter control by regional Internet registries over the allocation of addresses to local Internet registries
- Network renumbering to reclaim large blocks of address space allocated in the early days of the Internet

The primary address pool of the Internet, maintained by IANA, was exhausted on 3 February 2011 when the last 5 blocks were allocated to the 5 RIRs.[7] [8] APNIC was the first RIR to exhaust its regional pool on 15 April 2011,

except for a small amount of address space reserved for the transition to IPv6, which will be allocated under a much more restricted policy.[9]

The accepted and standardized solution is the migration to Internet Protocol Version 6. The address size was increased in IPv6 to 128 bits, providing a vastly increased address space that also allows improved route aggregation across the Internet and offers large subnetwork allocations of a minimum of 2^{64} host addresses to end-users. Migration to IPv6 is in progress but completion is expected to take considerable time.

Packet structure

An IP packet consists of a header section and a data section.

Header

The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional (red background in table) and aptly named: options. The fields in the header are packed with the most significant byte first (big endian), and for the diagram and discussion, the most significant bits are considered to come first (MSB 0 bit numbering). The most significant bit is numbered 0, so the version field is actually found in the four most significant bits of the first byte, for example.

bit offset	**0–3**	**4–7**	**8–13**	**14-15**	**16–18**	**19–31**
0	Version	Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live		Protocol		Header Checksum	
96	Source IP Address					
128	Destination IP Address					
160	Options (if Header Length > 5)					
160 or 192+	Data					

Version

The first header field in an IP packet is the four-bit version field. For IPv4, this has a value of 4 (hence the name IPv4).

Internet Header Length (IHL)

The second field (4 bits) is the Internet Header Length (IHL) telling the number of 32-bit words in the header. Since an IPv4 header may contain a variable number of options, this field specifies the size of the header (this also coincides with the offset to the data). The minimum value for this field is 5 (RFC 791), which is a length of 5×32 = 160 bits = 20 bytes. Being a 4-bit value, the maximum length is 15 words (15×32 bits) or 480 bits = 60 bytes.

Differentiated Services Code Point (DSCP)

Originally defined as the Type of Service field, this field is now defined by RFC 2474 for Differentiated services (DiffServ). New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP) that is used for interactive data voice exchange.

Explicit Congestion Notification (ECN)

Defined in RFC 3168 and allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.

Total Length

This 16-bit field defines the entire datagram size, including header and data, in bytes. The minimum-length datagram is 20 bytes (20-byte header + 0 bytes data) and the maximum is 65,535 bytes — the maximum value of a 16-bit word. The minimum size datagram that any host is required to be able to handle is 576 bytes, but most modern hosts handle much larger packets. Sometimes subnetworks impose further restrictions on the size, in which case datagrams must be fragmented. Fragmentation is handled in either the host or packet switch in IPv4.

Identification

This field is an identification field and is primarily used for uniquely identifying fragments of an original IP datagram. Some experimental work has suggested using the ID field for other purposes, such as for adding packet-tracing information to datagrams in order to help trace back datagrams with spoofed source addresses.[10]

Flags

A three-bit field follows and is used to control or identify fragments. They are (in order, from high order to low order):

- bit 0: Reserved; must be zero.[11]
- bit 1: Don't Fragment (DF)
- bit 2: More Fragments (MF)

If the DF flag is set and fragmentation is required to route the packet then the packet is dropped. This can be used when sending packets to a host that does not have sufficient resources to handle fragmentation. It can also be used for Path MTU Discovery, either automatically by the host IP software, or manually using diagnostic tools such as ping or traceroute.

For unfragmented packets, the MF flag is cleared. For fragmented packets, all fragments except the last have the MF flag set. The last fragment has a non-zero Fragment Offset field, differentiating it from an unfragmented packet.

Fragment Offset

The fragment offset field, measured in units of eight-byte blocks, is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of $(2^{13} - 1) \times 8 = 65{,}528$ bytes which would exceed the maximum IP packet length of 65,535 bytes with the header length included (65,528 + 20 = 65,548 bytes).

Time To Live (TTL)

An eight-bit time to live field helps prevent datagrams from persisting (e.g. going in circles) on an internet. This field limits a datagram's lifetime. It is specified in seconds, but time intervals less than 1 second are rounded up to 1. In latencies typical in practice, it has come to be a hop count field. Each router that a datagram crosses decrements the TTL field by one. When the TTL field hits zero, the packet is no longer forwarded by a packet switch and is discarded. Typically, an ICMP Time Exceeded message is sent back to the sender to inform it that the packet has been discarded. The reception of these ICMP messages is at the heart of how traceroute works.

Protocol

This field defines the protocol used in the data portion of the IP datagram. The Internet Assigned Numbers Authority maintains a list of IP protocol numbers which was originally defined in RFC 790.

Header Checksum

The 16-bit checksum field is used for error-checking of the header. At each hop, the checksum of the header must be compared to the value of this field. If a header checksum is found to be mismatched, then the packet is

discarded. Errors in the data field must be handled by the encapsulated protocol and both UDP and TCP have checksum fields.

As the TTL field is decremented on each hop, a new checksum must be computed each time. The method used to compute the checksum is defined by RFC 1071:

> *The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.*

For example, use Hex 4500003044224000800600008c7c19acae241e2b (20 bytes IP header):

> 4500 + 0030 + 4422 + 4000 + 8006 + 0000 + 8c7c + 19ac + ae24 + 1e2b = 2BBCF
>
> 2 + BBCF = BBD1 = 1011101111010001, the 1'S of sum = 0100010000101110 = 442E

To validate a header's checksum the same algorithm may be used - the checksum of a header which contains a correct checksum field is a word containing all zeros (value 0):

> 2BBCF + 442E = 2FFFD. 2 + FFFD = FFFF. the 1'S of FFFF = 0.

Source address

An IPv4 address indicating the sender of the packet. Note that this address may be changed in transit by a network address translation device.

Destination address

An IPv4 address indicating the receiver of the packet. As with the Source address, this may be changed in transit by a network address translation device.

Options

Additional header fields may follow the destination address field, but these are not often used. Note that the value in the IHL field must include enough extra 32-bit words to hold all the options (plus any padding needed to ensure that the header contains an integral number of 32-bit words). The list of options may be terminated with an EOL (End of Options List, 0x00) option; this is only necessary if the end of the options would not otherwise coincide with the end of the header. The possible options that can be put in the header are as follows:

Field	Size (bits)	Description
Copied	1	Set to 1 if the options need to be copied into all fragments of a fragmented packet.
Option Class	2	A general options category. 0 is for "*control*" options, and 2 is for "*debugging and measurement*". 1, and 3 are reserved.
Option Number	5	Specifies an option.
Option Length	8	Indicates the size of the entire option (including this field). This field may not exist for simple options.
Option Data	Variable	Option-specific data. This field may not exist for simple options.

- Note: If the header length is greater than 5, i.e. it is from 6 to 15, it means that the options field is present and must be considered.
- Note: Copied, Option Class, and Option Number are sometimes referred to as a single eight-bit field - the *Option Type*.

The use of the LSRR and SSRR options (Loose and Strict Source and Record Route) is discouraged because they create security concerns; many routers block packets containing these options.

Data

The data portion of the packet is not included in the packet checksum. Its contents are interpreted based on the value of the Protocol header field.

In a typical IP implementation, standard protocols such as TCP and UDP are implemented in the OS kernel for performance reasons. Other protocols such as ICMP may be partially implemented by the kernel, or implemented purely in user software. Protocols not implemented in-kernel, and not exposed by standard APIs such as BSD sockets, are typically implemented using a 'raw socket' API.

Some of the common protocols for the data portion are listed below:

Protocol Number	Protocol Name	Abbreviation
1	Internet Control Message Protocol	ICMP
2	Internet Group Management Protocol	IGMP
6	Transmission Control Protocol	TCP
17	User Datagram Protocol	UDP
41	IPv6 encapsulation	ENCAP
89	Open Shortest Path First	OSPF
132	Stream Control Transmission Protocol	SCTP

See List of IP protocol numbers for a complete list.

Fragmentation and reassembly

The Internet Protocol is the facility in the Internet architecture that enables different networks to exchange traffic and route traffic across one another. The design accommodates networks of diverse physical nature; it is independent of the underlying transmission technology used in the Link Layer. Link Layer networks of different hardware design usually vary not only in transmission speed, but also in the structure and size of valid framing methods, characterized by the maximum transmission unit (MTU) parameter. To fulfill the role of IP to traverse networks, it was necessary to implement a mechanism to automatically adjust the size of transmission units to adapt to the underlying technology. This introduced the need for fragmentation of IP datagrams. In IPv4, this function was placed at the Internet Layer, and is performed in IPv4 routers, which thus only require this layer as highest one implemented in their design.

In contrast, the next generation of the Internet Protocol, namely IPv6, does not require routers to perform fragmentation; instead, hosts must determine the path maximum transmission unit in advance of transmission and send conforming datagrams.

Fragmentation

When a device receives an IP packet it examines the destination address and determines the outgoing interface to use. This interface has an associated MTU that dictates the maximum data size for its payload. If the data size is bigger than the MTU then the device must fragment the data.

The device then segments the data into segments where each segment is less-than-or-equal-to the MTU less the IP header size (20 bytes minimum; 60 bytes maximum). Each segment is then put into its own IP packet with the following changes:

- The *total length* field is adjusted to the segment size
- The *more fragments* (MF) flag is set for all segments except the last one, which is set to 0

- The *fragment offset* field is set accordingly based on the offset of the segment in the original data payload. This is measured in units of eight-byte blocks.
- The *header checksum* field is recomputed.

For example, for an IP header of length 20 bytes and an Ethernet MTU of 1,500 bytes the fragment offsets would be: 0, (1480/8) = 185, (2960/8) = 370, (4440/8) = 555, (5920/8) = 740, etc.

By some chance if a packet changes link layer protocols or the MTU reduces then these fragments would be fragmented again.

For example, if a 4,500-byte data payload is inserted into an IP packet with no options (thus total length is 4,520 bytes) and is transmitted over a link with an MTU of 2,500 bytes then it will be broken up into two fragments:

#	Total length		More fragments (MF) flag set?	Fragment offset
	Header	Data		
1	2500		Yes	0
	20	2480		
2	2040		No	310
	20	2020		

Now, let's say the MTU drops to 1,500 bytes. Each fragment will individually be split up into two more fragments each:

#	Total length		More fragments (MF) flag set?	Fragment offset
	Header	Data		
1	1500		Yes	0
	20	1480		
2	1020		Yes	185
	20	1000		
3	1500		Yes	310
	20	1480		
4	560		No	495
	20	540		

Indeed, the amount of data has been preserved — 1480 + 1000 + 1480 + 540 = 4500 — and the last fragment offset (495) * 8 (bytes) plus data — 3960 + 540 = 4500 — is also the total length.

Note that fragments 3 & 4 were derived from the original fragment 2. When a device must fragment the last fragment then it must set the flag for all but the last fragment it creates (fragment 4 in this case). Last fragment would be set to 0 value.

Reassembly

When a receiver detects an IP packet where either of the following is true:

- "more fragments" flag set
- "fragment offset" field is non-zero

then the receiver knows the packet is a fragment. The receiver then stores the data with the identification field, fragment offset, and the more fragments flag. When the receiver receives a fragment with the more fragments flag set to 0 then it knows the length of the original data payload since the fragment offset multiplied by 8 (bytes) plus the data length is equivalent to the original data payload size.

Using the example above, when the receiver receives fragment 4 the fragment offset (495 or 3960 bytes) and the data length (540 bytes) added together yield 4500 — the original data length.

Once it has all the fragments then it can reassemble the data in proper order (by using the fragment offsets) and pass it up the stack for further processing.

Assistive protocols

The Internet Protocol is the protocol that defines and enables internetworking at the Internet Layer and thus forms the Internet. It uses a logical addressing system. IP addresses are not tied in any permanent manner to hardware identifications and, indeed, a network interface can have multiple IP addresses. Hosts and routers need additional mechanisms to identify the relationship between device interfaces and IP addresses, in order to properly deliver an IP packet to the destination host on a link. The Address Resolution Protocol (ARP) performs this IP address to hardware address (MAC address) translation for IPv4. In addition, the reverse correlation is often necessary. For example, when an IP host is booted or connected to a network it needs to determine its IP address, unless an address is preconfigured by an administrator. Protocols for such inverse correlations exist in the Internet Protocol Suite. Currently used methods are Dynamic Host Configuration Protocol (DHCP), Bootstrap Protocol (BOOTP) and, infrequently, reverse ARP.

Notes

[1] "INET(3) man page" (http://www.unix.com/man-page/Linux/3/inet_addr/). . Retrieved 2010-11-28.
[2] http://technet.microsoft.com/en-us/library/cc779089%28WS.10%29.aspx
[3] http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf
[4] Robert Braden (October 1989). "Requirements for Internet Hosts -- Communication Layers" (http://tools.ietf.org/html/rfc1122#page-31). IETF. p. 31. RFC 1122. .
[5] Robert Braden (October 1989). "Requirements for Internet Hosts -- Communication Layers" (http://tools.ietf.org/html/rfc1122#page-66). IETF. p. 66. RFC 1122. .
[6] "World 'running out of Internet addresses'" (http://technology.inquirer.net/infotech/infotech/view/20110121-315808/World-running-out-of-Internet-addresses). . Retrieved 2011-01-23.
[7] Smith, Lucie; Lipner, Ian (3 February 2011). "Free Pool of IPv4 Address Space Depleted" (http://www.nro.net/news/ipv4-free-pool-depleted). Number Resource Organization. . Retrieved 3 February 2011.
[8] ICANN,nanog mailing list. "Five /8s allocated to RIRs - no unallocated IPv4 unicast /8s remain" (http://mailman.nanog.org/pipermail/nanog/2011-February/032107.html). .
[9] Asia-Pacific Network Information Centre (15 April 2011). "APNIC IPv4 Address Pool Reaches Final /8" (http://www.apnic.net/publications/news/2011/final-8). . Retrieved 15 April 2011.
[10] Savage, Stefan. "Practical network support for IP traceback" (http://portal.acm.org/citation.cfm?id=347057.347560). . Retrieved 2010-09-06.
[11] As an April Fools' joke, proposed for use in RFC 3514 as the "Evil bit".

References

External links

- RFC 791 — Internet Protocol
- http://www.iana.org — Internet Assigned Numbers Authority (IANA)
- http://www.networksorcery.com/enp/protocol/ip.htm — IP Header Breakdown, including specific options
- RFC 3344 — IPv4 Mobility
- IPv6 vs. carrier-grade NAT/squeezing more out of IPv4 (http://www.networkworld.com/news/2010/060710-tech-argument-ipv6-nat.html)

Address exhaustion:

- RIPE report on address consumption as of October 2003 (http://www.ripe.net/rs/news/ipv4-ncc-20031030.html)
- Official current state of IPv4 /8 allocations, as maintained by IANA (http://www.iana.org/assignments/ipv4-address-space)
- Dynamically generated graphs of IPv4 address consumption with predictions of exhaustion dates — Geoff Huston (http://www.potaroo.net/tools/ipv4/index.html)
- IP addressing in China and the myth of address shortage (http://www.apnic.net/community/about-the-internet-community/internet-governance/articles/ip-addressing-in-china-2004)
- Countdown of remaining IPv4 available addresses (http://www.inetcore.com/project/ipv4ec/index_en.html) (estimated)

IPv4 subnetting reference

In the IPv4 address space certain address blocks are specially allocated or reserved for special uses such as loopback interfaces, private networks (RFC 1918)[1] , and state-less autoconfiguration (Zeroconf, RFC 3927)[2] of interfaces. Such addresses may be used without registration or allocation from Regional Internet Registries (RIRs). However, these address ranges must not be routed into the public Internet infrastructure.

The netmask is a bitmask that can be used to separate the bits of the network identifier from the bits of the host identifier. It is often written in the same notation used to denote IP addresses.

Not all sizes of prefix announcement may be routable on the public Internet: see routing, peering.

Class	Leading bits	Start	End	Default Subnet Mask in dotted decimal	CIDR notation
A	0	0.0.0.0	127.255.255.255	255.0.0.0	/8
B	10	128.0.0.0	191.255.255.255	255.255.0.0	/16
C	110	192.0.0.0	223.255.255.255	255.255.255.0	/24
D	1110	224.0.0.0	239.255.255.255	not defined	not defined
E	1111	240.0.0.0	255.255.255.254	not defined	not defined

The blocks numerically at the start and end of classes A, B and C were originally reserved for special addressing or future features, i.e., 0.0.0.0/8 and 127.0.0.0/8 are reserved in former class A; 128.0.0.0/16 and 191.255.0.0/16 were reserved in former class B but are now available for assignment; 192.0.0.0/24 and 223.255.255.0/24 are reserved in former class C.

While the 127.0.0.0/8 network is a Class A network, it is designated for loopback and cannot be assigned to a network.

See (RFC 5735 Special Use IPv4 Addresses)[3]

CIDR	host bits	Netmask	Hosts in subnet	Classful name	Typical usage
/8	24	255.0.0.0	16777216 = 2^{24}	Class A (see this list)	Largest block allocation made by IANA
/9	23	255.128.0.0	8388608 = 2^{23}		
/10	22	255.192.0.0	4194304 = 2^{22}		
/11	21	255.224.0.0	2097152 = 2^{21}		
/12	20	255.240.0.0	1048576 = 2^{20}		
/13	19	255.248.0.0	524288 = 2^{19}		
/14	18	255.252.0.0	262144 = 2^{18}		
/15	17	255.254.0.0	131072 = 2^{17}		
/16	16	255.255.0.0	65536 = 2^{16}	Class B	
/17	15	255.255.128.0	32768 = 2^{15}		ISP / large business
/18	14	255.255.192.0	16384 = 2^{14}		ISP / large business
/19	13	255.255.224.0	8192 = 2^{13}		ISP / large business
/20	12	255.255.240.0	4096 = 2^{12}		Small ISP / large business
/21	11	255.255.248.0	2048 = 2^{11}		Small ISP / large business
/22	10	255.255.252.0	1024 = 2^{10}		
/23	9	255.255.254.0	512 = 2^{9}		
/24	8	255.255.255.0	256 = 2^{8}	Class C	Large LAN
/25	7	255.255.255.128	128 = 2^{7}		Large LAN
/26	6	255.255.255.192	64 = 2^{6}		Small LAN
/27	5	255.255.255.224	32 = 2^{5}		Small LAN
/28	4	255.255.255.240	16 = 2^{4}		Small LAN
/29	3	255.255.255.248	8 = 2^{3}		Smallest multi-host network
/30	2	255.255.255.252	4 = 2^{2}		"Glue network" (point to point links)
/31	1	255.255.255.254	2 = 2^{1}		Rarely used, point to point links (RFC 3021)
/32	0	255.255.255.255	1 = 2^{0}		Host route

In common usage, the "host all zeros" address is reserved for referring to the entire network, while the "host all ones" address is used as a broadcast address in the given subnet; this reduces the number of addresses available for hosts by 2. This explains the reference to /31 networks as "Rarely Used," as the only possible addresses on a /31 network are "host all ones" and "host all zeros." RFC 3021 creates an exception to the "host all ones" and "host all zeros" broadcast usage to make /31 networks usable for point-to-point links. In practice, however, point-to-point links are still typically implemented using /30 networks, or occasionally by /32 and point-to-point explicit host routes. There is generally no technical advantage to /31 versus /32, although one or the other may be more convenient based on

other issues. A /30 is always wasteful and has as its sole advantage that it behaves "as expected" for any other subnetwork.

Notes

[1] RFC 1918 http://www.rfc-editor.org/rfc/rfc1918.txt
[2] RFC 3927 http://www.rfc-editor.org/rfc/rfc3927.txt
[3] RFC 5735 http://www.rfc-editor.org/rfc/rfc5735.txt

IPv6

Internet Protocol version 6 (**IPv6**) is a version of the Internet Protocol (IP). It is designed to succeed the Internet Protocol version 4 (IPv4). The Internet operates by transferring data between hosts in small packets that are independently routed across networks as specified by an international communications protocol known as the Internet Protocol.

Each host or computer on the Internet requires an IP address in order to communicate. The growth of the Internet has created a need for more addresses than are possible with IPv4. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with this long-anticipated IPv4 address exhaustion, and is described in Internet standard document RFC 2460, published in December 1998.[1] Like IPv4, IPv6 is an internet-layer protocol for packet-switched internetworking and provides end-to-end datagram transmission across multiple IP networks. While IPv4 allows 32 bits for an Internet Protocol address, and can therefore support 2^{32} (4,294,967,296) addresses, IPv6 uses 128-bit addresses, so the new address space supports 2^{128} (approximately 340 undecillion or 3.4×10^{38}) addresses. This expansion allows for many more devices and users on the internet as well as extra flexibility in allocating addresses and efficiency for routing traffic. It also eliminates the primary need for network address translation (NAT), which gained widespread deployment as an effort to alleviate IPv4 address exhaustion.

IPv6 also implements additional features not present in IPv4. It simplifies aspects of address assignment (stateless address autoconfiguration), network renumbering and router announcements when changing Internet connectivity providers. The IPv6 subnet size has been standardized by fixing the size of the host identifier portion of an address to 64 bits to facilitate an automatic mechanism for forming the host identifier from link-layer media addressing information (MAC address). Network security is also integrated into the design of the IPv6 architecture, and the IPv6 specification mandates support for IPsec as a fundamental interoperability requirement.

The last top level (`/8`) block of free IPv4 addresses was assigned in February 2011 by IANA to the 5 RIRs, although many free addresses still remain in most assigned blocks and each RIR will continue with standard policy until it is at its last `/8`. After that, only 1024 addresses (a `/22`) are made available from the RIR for each LIR – currently, only APNIC has already reached this stage.[2] While IPv6 is supported on all major operating systems in use in commercial, business, and home consumer environments,[3] IPv6 does not implement interoperability features with IPv4, and creates essentially a parallel, independent network. Exchanging traffic between the two networks requires special translator gateways, but modern computer operating systems implement dual-protocol software for transparent access to both networks either natively or using a tunneling protocol such as 6to4, 6in4, or Teredo. In December 2010, despite marking its 12th anniversary as a Standards Track protocol, IPv6 was only in its infancy in terms of general worldwide deployment. A 2008 study[4] by Google Inc. indicated that penetration was still less than one percent of Internet-enabled hosts in any country at that time.

Motivation and origins

IPv4

The first publicly used version of the Internet Protocol, Version 4 (IPv4), provides an addressing capability of 2^{32} or approximately 4.3 billion addresses. Address exhaustion was not initially a concern in IPv4 as this version was originally presumed to be an internal test within ARPA, and not intended for public use.

> The decision to put a 32-bit address space on there was the result of a year's battle among a bunch of engineers who couldn't make up their minds about 32, 128, or variable-length. And after a year of fighting, I said--I'm now at ARPA, I'm running the program, I'm paying for this stuff, I'm using American tax dollars, and I wanted some progress because we didn't know if this was going to work. So I said: OK, it's 32-bits. That's enough for an experiment; it's 4.3 billion terminations. Even the Defense Department doesn't need 4.3 billion of everything and couldn't afford to buy 4.3 billion edge devices to do a test anyway. So at the time I thought we were doing an experiment to prove the technology and that if it worked we'd have opportunity to do a production version of it. Well, it just escaped! It got out and people started to use it, and then it became a commercial thing. So this [IPv6] is the production attempt at making the network scalable.
>
> —Vint Cerf, Google IPv6 Conference 2008 [5]

During the first decade of operation of the Internet (by the late 1980s), it became apparent that methods had to be developed to conserve address space. In the early 1990s, even after the redesign of the addressing system using a classless network model, it became clear that this would not suffice to prevent IPv4 address exhaustion, and that further changes to the Internet infrastructure were needed.[6]

Working-group proposal

By the beginning of 1992, several proposals appeared and by the end of 1992, the IETF announced a call for white papers.[7] In September 1993, the IETF created a temporary, ad-hoc *IP Next Generation* (IPng) area to deal specifically with IPng issues. The new area was led by Allison Mankin and Scott Bradner, and had a directorate with 15 engineers from diverse backgrounds for direction-setting and preliminary document review:[6] [8] [9]

The Internet Engineering Task Force adopted the IPng model on July 25, 1994, with the formation of several IPng working groups.[6] By 1996, a series of RFCs was released defining Internet Protocol version 6 (IPv6), starting with RFC 1883. (Version 5 was used by the experimental Internet Stream Protocol.)

It is widely expected that IPv4 will be supported alongside IPv6 for the foreseeable future. IPv4-only and IPv6-only nodes cannot communicate directly, and need assistance from an intermediary gateway or must use other transition mechanisms.

Exhaustion of IPv4 addresses

On February 3, 2011, in a ceremony in Miami, the Internet Assigned Numbers Authority (IANA) assigned the last batch of 5 `/8` address blocks to the Regional Internet Registries.,[10] officially depleting the global pool of completely fresh blocks of addresses.[11] Each of the address blocks represents approximately 16.7 million possible addresses, or over 80 million combined potential addresses.

These addresses could well be fully consumed within three to six months of that time at current rates of allocation.[12] APNIC was the first RIR to exhaust its regional pool on 15 April 2011, except for a small amount of address space reserved for the transition to IPv6, which will be allocated in a much more restricted way.[13]

In 2003, the director of Asia-Pacific Network Information Centre (APNIC), Paul Wilson, stated that, based on then-current rates of deployment, the available space would last for one or two decades.[14] In September 2005, a report by Cisco Systems suggested that the pool of available addresses would exhaust in as little as 4 to 5 years.[15] In 2008, a policy process started for the end-game and post-exhaustion era.[16] In 2010, a daily updated report

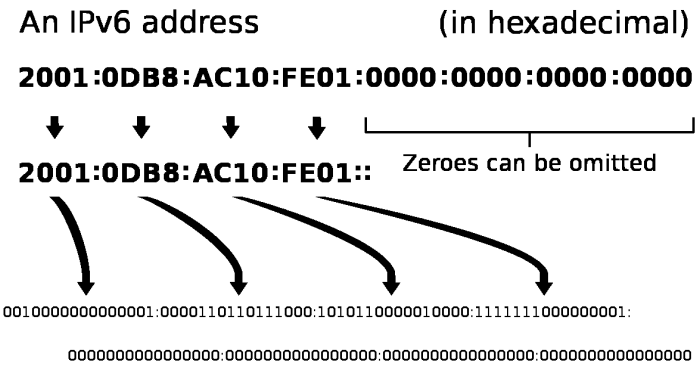
projected the global address pool exhaustion by the first quarter of 2011, and depletion at the five regional Internet registries before the end of 2011.[17]

Comparison to IPv4

IPv6 specifies a new packet format, designed to minimize packet header processing by routers.[1] [18] Because the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable. However, in most respects, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP and NTPv3.

Larger address space

The most important feature of IPv6 is a much larger address space than in IPv4. The length of an IPv6 address is 128 bits, compared to 32 bits in IPv4.[1] The address space therefore supports 2^{128} or approximately 3.4×10^{38} addresses. By comparison, this amounts to approximately 4.8×10^{28} addresses for each of the seven billion people alive in 2011.[19] In addition, the IPv4 address space is poorly allocated, with approximately 14% of all available addresses utilized.[20] While these numbers are large, it was not the intent of the designers of the IPv6 address space to assure geographical saturation with usable addresses. Rather, the longer addresses simplify allocation of addresses, enable efficient route aggregation, and allow implementation of special addressing features. In IPv4, complex Classless Inter-Domain Routing (CIDR) methods were developed to make the best use of the small address space. The standard size of a subnet in IPv6 is 2^{64} addresses, the square of the size of the entire IPv4 address space. Thus, actual address space utilization rates will be small in IPv6, but network management and routing efficiency is improved by the large subnet space and hierarchical route aggregation.



Decomposition of an IPv6 address into its binary form

Renumbering an existing network for a new connectivity provider with different routing prefixes is a major effort with IPv4.[21] [22] With IPv6, however, changing the prefix announced by a few routers can in principle renumber an entire network since the host identifiers (the least-significant 64 bits of an address) can be independently self-configured by a host.

Multicasting

Multicasting, the transmission of a packet to multiple destinations in a single send operation, is part of the base specification in IPv6. In IPv4 this is an optional although commonly implemented feature.[23] IPv6 multicast addressing shares common features and protocols with IPv4 multicast, but also provides changes and improvements by eliminating the need for certain protocols. IPv6 does not implement traditional IP broadcast, i.e. the transmission of a packet to all hosts on the attached link using a special *broadcast address*, and therefore does not define broadcast addresses. In IPv6, the same result can be achieved by sending a packet to the link-local *all nodes* multicast group at address `ff02::1`, which is analogous to IPv4 multicast to address `224.0.0.1`. IPv6 also supports new multicast solutions, including embedding rendezvous point addresses in an IPv6 multicast group address which simplifies the deployment of inter-domain solutions.[24]

In IPv4 it was very difficult for an organization to get even one globally routable multicast group assignment and the implementation of inter-domain solutions was very arcane.[25] Unicast address assignments by a local Internet registry for IPv6 have at least a 64-bit routing prefix, yielding the smallest subnet size available in IPv6 (also 64 bits). With such an assignment it is possible to embed the unicast address prefix into the IPv6 multicast address format, while still providing a 32-bit block, the least significant bits of the address, or approximately 4.2 billion multicast group identifiers. Thus each user of an IPv6 subnet automatically has available a set of globally routable source-specific multicast groups for multicast applications.[26]

Stateless address autoconfiguration (SLAAC)

IPv6 hosts can configure themselves automatically when connected to a routed IPv6 network using Internet Control Message Protocol version 6 (ICMPv6) router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters; if configured suitably, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters.[27]

If IPv6 stateless address autoconfiguration is unsuitable for an application, a network may use stateful configuration with the Dynamic Host Configuration Protocol version 6 (DHCPv6) or hosts may be configured statically.

Routers present a special case of requirements for address configuration, as they often are sources for autoconfiguration information, such as router and prefix advertisements. Stateless configuration for routers can be achieved with a special router renumbering protocol.[28]

Mandatory support for network-layer security

Internet Protocol Security (IPsec) was originally developed for IPv6, but found widespread deployment first in IPv4, into which it was back-engineered. IPsec is an integral part of the base protocol suite in IPv6.[1] IPsec support is mandatory in IPv6[29] but optional for IPv4.

Simplified processing by routers

In IPv6, the packet header and the process of packet forwarding have been simplified. Although IPv6 packet headers are at least twice the size of IPv4 packet headers, packet processing by routers is generally more efficient,[1] [18] thereby extending the end-to-end principle of Internet design. Specifically:

- The packet header in IPv6 is simpler than that used in IPv4, with many rarely used fields moved to separate optional header extensions.
- IPv6 routers do not perform fragmentation. IPv6 hosts are required to either perform path MTU discovery, perform end-to-end fragmentation, or to send packets no larger than the IPv6 default minimum MTU size of 1280 octets.
- The IPv6 header is not protected by a checksum; integrity protection is assumed to be assured by both link-layer and higher-layer (TCP, UDP, etc.) error detection.[30] Therefore, IPv6 routers do not need to recompute a checksum when header fields (such as the time to live (TTL) or hop count) change.[31]
- The *TTL* field of IPv4 has been renamed to *Hop Limit*, reflecting the fact that routers are no longer expected to compute the time a packet has spent in a queue.

Mobility

Unlike mobile IPv4, mobile IPv6 avoids triangular routing and is therefore as efficient as native IPv6. IPv6 routers may also support network mobility which allows entire subnets to move to a new router connection point without renumbering.[32]

Options extensibility

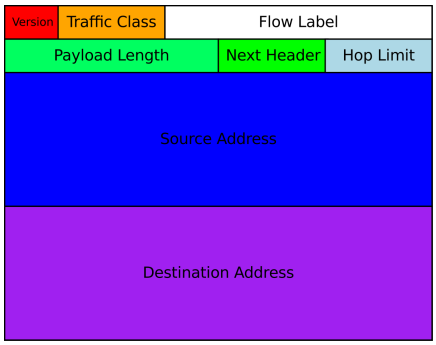
The IPv6 protocol header has a fixed size (40 octets). Options are implemented as additional extension headers after the IPv6 header, which limits their size only by the size of an entire packet. The extension header mechanism provides extensibility to support future services for quality of service, security, mobility, and others, without redesign of the basic protocol.[1]

Jumbograms

IPv4 limits packets to 65535 ($2^{16} - 1$) octets of payload. IPv6 has optional support for packets over this limit, referred to as jumbograms, which can be as large as 4294967295 ($2^{32} - 1$) octets. The use of jumbograms may improve performance over high-MTU links. The use of jumbograms is indicated by the Jumbo Payload Option header.[33]

Packet format

The IPv6 packet is composed of two parts: the packet header and the payload. The header consists of a fixed portion with minimal functionality required for all packets and may contain optional extension to implement special features.



IPv6 packet header.

The fixed header occupies the first 40 octets (320 bits) of the IPv6 packet. It contains the source and destination addresses, traffic classification options, a hop counter, and a pointer for extension headers if any. The *Next Header* field, present in each extension as well, points to the next element in the chain of extensions. The last field points to the upper-layer protocol that is carried in the packet's payload.

Extension headers carry options that are used for special treatment of a packet in the network, e.g., for routing, fragmentation, and for security using the IPsec framework.

The payload can have a size of up to 64KB without special options, or larger with a jumbo payload option in a *Hop-By-Hop Options* extension header.

Unlike in IPv4, fragmentation is handled only in the end points of a communication session; routers never fragment a packet, and hosts are expected to use Path MTU Discovery to select a packet size that can traverse the entire communications path.

Addressing

The most important feature of IPv6 is a much larger address space than in IPv4. IPv6 addresses are 128 bits long, compared to only 32 bits previously.[34] While the IPv4 address space contains only about 4.3×10^9 (4.3 billion) addresses, IPv6 supports approximately 3.4×10^{38} (340 undecillion) unique addresses, deemed enough for the foreseeable future.[35]

IPv6 addresses are written in eight groups of four hexadecimal digits separated by colons, for example, `2001:0db8:85a3:0000:0000:8a2e:0370:7334`. IPv6 unicast addresses other than those that start with binary 000 are logically divided into two parts: a 64-bit (sub-)network prefix, and a 64-bit interface identifier.[36]

For stateless address autoconfiguration (SLAAC) to work, subnets require a /64 address block as defined in RFC 4291 section 2.5.1. Local Internet registries get assigned at least /32 blocks, which they divide among ISPs.[37] The obsolete RFC 3177 recommended the assignment of a /48 to end consumer sites. This was replaced by RFC 6177, which "recommends giving home sites significantly more than a single /64, but does not recommend that every home site be given a /48 either." /56s are specifically considered. It remains to be seen if ISPs will honor this recommendation; for example, during initial trials Comcast customers have been given a single /64 network.[38]

IPv6 addresses are classified by three types of networking methodologies: unicast addresses identify each network interface, anycast addresses identify a group of interfaces, usually at different locations of which the nearest one is automatically selected, and multicast addresses are used to deliver one packet to many interfaces. The broadcast method is not implemented in IPv6. Each IPv6 address has a scope, which specifies in which part of the network it is valid and unique. Some addresses are unique only on the local (sub-)network; Others are globally unique.

Some IPv6 addresses are reserved for special purposes, such as the address for loopback, 6to4 tunneling, Teredo tunneling and several more. See RFC 5156. Also, some address ranges are considered special, such as link-local addresses for use on the local link only, Unique Local addresses (ULA) as described in RFC 4193 and solicited-node multicast addresses used in the Neighbor Discovery Protocol.

IPv6 in the Domain Name System

In the Domain Name System, hostnames are mapped to IPv6 addresses by *AAAA* resource records, so-called *quad-A* records. For reverse resolution, the IETF reserved the domain `ip6.arpa`, where the name space is hierarchically divided by the 1-digit hexadecimal representation of nibble units (4 bits) of the IPv6 address. This scheme is defined in RFC 3596.

Address Format

IPv6 addresses have two logical parts: a 64-bit network prefix, and a 64-bit host address part. (The host address is often automatically generated from the interface MAC address.[39]) An IPv6 address is represented by 8 groups of 16-bit hexadecimal values separated by colons (:) shown as follows:

```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

The hexadecimal digits are case-insensitive.

The 128-bit IPv6 address can be abbreviated with the following rules:

- Rule one: Leading zeroes within a 16-bit value may be omitted. For example, the address `fe80:0000:0000:0000:0202:b3ff:fe1e:8329` may be written as `fe80:0:0:0:202:b3ff:fe1e:8329`
- Rule two: One group of consecutive zeroes within an address may be replaced by a double colon. For example, `fe80:0:0:0:202:b3ff:fe1e:8329` becomes `fe80::202:b3ff:fe1e:8329`

A single IPv6 address can be represented in several different ways, such as `2001:db8::1:0:0:1` and `2001:0DB8:0:0:1::1`. RFC 5952 recommends a canonical textual representation.

Transition mechanisms

Until IPv6 completely supplants IPv4, a number of transition mechanisms[40] are needed to enable IPv6-only hosts to reach IPv4 services and to allow isolated IPv6 hosts and networks to reach the IPv6 Internet over the IPv4 infrastructure.

For the period while IPv6 hosts and routers co-exist with IPv4 systems various proposals have been made:

- RFC 2185, *Routing Aspects of IPv6 Transition*
- RFC 2766, *Network Address Translation — Protocol Translation NAT-PT*, obsoleted as explained in RFC 4966 *Reasons to Move the Network Address Translator — Protocol Translator NAT-PT to Historic Status*
- RFC 3053, *IPv6 Tunnel Broker*
- RFC 3056, *6to4. Connection of IPv6 Domains via IPv4 Clouds*
- RFC 3142, *An IPv6-to-IPv4 Transport Relay Translator*
- RFC 4213, *Basic Transition Mechanisms for IPv6 Hosts and Routers*
- RFC 4380, *Teredo: Tunneling IPv6 over UDP through Network Address Translations NATs*
- RFC 4798, *Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)*
- RFC 5214, *Intra-Site Automatic Tunnel Addressing Protocol ISATAP*
- RFC 5569, *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*
- RFC 5572, *IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)*
- RFC 6180, *Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment*
- RFC 6343, *Advisory Guidelines for 6to4 Deployment*

Dual IP stack implementation

The dual-stack protocol implementation in an operating system is a fundamental IPv4-to-IPv6 transition technology. It implements IPv4 and IPv6 protocol stacks either independently or in a hybrid form. The hybrid form is commonly implemented in modern operating systems supporting IPv6. Dual-stack hosts are described in RFC 4213.

Modern hybrid dual-stack implementations of IPv4 and IPv6 allow programmers to write networking code that works transparently on IPv4 or IPv6. The software may use hybrid sockets designed to accept both IPv4 and IPv6 packets. When used in IPv4 communications, hybrid stacks use an IPv6 application programming interface and represent IPv4 addresses in a special address format, the IPv4-mapped IPv6 address.

IPv4-mapped IPv6 addresses

Hybrid dual-stack IPv6/IPv4 implementations support a special class of addresses, the IPv4-mapped IPv6 addresses. This address type has its first 80 bits set to zero and the next 16 set to one, while its last 32 bits are filled with the IPv4 address. These addresses are commonly represented in the standard IPv6 format, but having the last 32 bits written in the customary dot-decimal notation of IPv4; for example, `::ffff:192.0.2.128` represents the IPv4 address `192.0.2.128`. It substitutes the old and deprecated IPv4-compatible IPv6 address formed by `::192.0.2.128`.[41]

Because of the significant internal differences between IPv4 and IPv6, some of the lower level functionality available to programmers in the IPv6 stack do not work identically with IPv4 mapped addresses. Some common IPv6 stacks do not support the IPv4-mapped address feature, either because the IPv6 and IPv4 stacks are separate implementations (e.g., Microsoft Windows 2000, XP, and Server 2003), or because of security concerns (OpenBSD) .[42] On these operating systems, it is necessary to open a separate socket for each IP protocol that is to be supported. On some systems, e.g., the Linux kernel, NetBSD, and FreeBSD, this feature is controlled by the socket option `IPV6_V6ONLY` as specified in RFC 3493.[43]

Tunneling

In order to reach the IPv6 Internet, an isolated host or network must use the existing IPv4 infrastructure to carry IPv6 packets. This is done using a technique known as *tunneling* which consists of encapsulating IPv6 packets within IPv4, in effect using IPv4 as a link layer for IPv6.

The direct encapsulation of IPv6 datagrams within IPv4 packets is indicated by IP protocol number 41. IPv6 can also be encapsulated within UDP packets e.g. in order to cross a router or NAT device that blocks protocol 41 traffic. Other encapsulation schemes, such as used in AYIYA or GRE, are also popular.

Conversely, on IPv6-only internet links, when access to IPv4 network facilities are needed, tunneling of IPv4 over IPv6 protocol occurs, using the IPv6 as a link layer for IPv4.

Automatic tunneling

Automatic tunneling refers to a technique where the routing infrastructure automatically determines the tunnel endpoints.

6to4 is recommended by RFC 3056 tunneling method for automatic tunneling, which uses protocol 41 encapsulation.[44] Tunnel endpoints are determined by using a well-known IPv4 anycast address on the remote side, and embedding IPv4 address information within IPv6 addresses on the local side. 6to4 is widely deployed today.

Teredo is an automatic tunneling technique that uses UDP encapsulation and can allegedly cross multiple NAT boxes.[45] IPv6, including 6to4 and Teredo tunneling, are enabled by default in Windows Vista[46] and Windows 7. Most Unix systems only implement native support for 6to4, but Teredo can be provided by third-party software such as Miredo.

ISATAP[47] treats the IPv4 network as a virtual IPv6 local link, with mappings from each IPv4 address to a link-local IPv6 address. Unlike 6to4 and Teredo, which are *inter-site* tunnelling mechanisms, ISATAP is an *intra-site* mechanism, meaning that it is designed to provide IPv6 connectivity between nodes within a single organisation.

Configured and automated tunneling (6in4)

In *configured tunneling*, the tunnel endpoints are explicitly configured, either by an administrator manually or the operating system's configuration mechanisms, or by an automatic service known as a tunnel broker;[48] this is also referred to as *automated tunneling*. Configured tunneling is usually more deterministic and easier to debug than automatic tunneling, and is therefore recommended for large, well-administered networks. Automated tunneling provides a compromise between the ease of use of automatic tunneling and the deterministic behaviour of configured tunneling.

Raw encapsulation of IPv6 packets using IPv4 protocol number 41 is recommended for configured tunneling; this is sometimes known as 6in4 tunneling. As with automatic tunneling, encapsulation within UDP may be used in order to cross NAT boxes and firewalls.

Proxying and translation for IPv6-only hosts

After the regional Internet registries have exhausted their pools of available IPv4 addresses, it is likely that hosts newly added to the Internet might only have IPv6 connectivity. For these clients to have backward-compatible connectivity to existing IPv4-only resources, suitable IPv6 transition mechanisms must be deployed.

One form of address translation is the use of a dual-stack application-layer proxy server, for example a web proxy.

NAT-like techniques for application-agnostic translation at the lower layers in routers and gateways have been proposed. The NAT-PT standard was dropped due to a number of criticisms,[49] however more recently the continued low adoption of IPv6 has prompted a new standardization effort under the name NAT64.

Application transition

RFC 4038, *Application Aspects of IPv6 Transition*, is an informational RFC that covers the topic of IPv4 to IPv6 application transition mechanisms. Other RFCs that pertain IPv6 at the application level are:

- RFC 3493, *Basic Socket Interface Extensions for IPv6*
- RFC 3542, *Advanced Sockets Application Program Interface (API) for IPv6*

Similar to the OS-level WAN stack, applications can be:

- IPv4 only
- IPv6 only
- dual set of IPv4 and IPv6 only
- hybrid IPv4 and IPv6

IPv6 readiness

Compatibility with IPv6 networking is mainly a software or firmware issue. However, much of the older hardware that could in principle be upgraded is likely to be replaced instead. The American Registry for Internet Numbers (ARIN) suggests that all Internet servers be prepared to serve IPv6-only clients by January 2012.[50] [51]

Software

Most personal computers running recent operating system versions are IPv6-ready. Most popular applications with network capabilities are ready, and most others could be easily upgraded with support from the developers. Java applications adhering to Java 1.4 (February 2002) standards have support for IPv6.[52]

Hardware and embedded systems

Low-level equipment like network adapters and network switches may not be affected by the change, since they transmit link-layer frames without inspecting the contents. Networking devices that obtain IP addresses or perform routing based on IP address do need IPv6 support.

Most equipment would be IPv6 capable with a software or firmware update if the device has sufficient storage and memory space for the new IPv6 stack. However, manufacturers may be reluctant to spend on software development costs for hardware they have already sold when they are poised for new sales from IPv6-ready equipment.

In some cases, non-compliant equipment needs to be replaced because the manufacturer no longer exists or software updates are not possible, for example, because the network stack is implemented in permanent read-only memory.

Consumers view networking devices as household appliances that do not need maintenance. Little effort has been made to educate consumers about the need to upgrade.

The CableLabs consortium published the 160 Mbit/s DOCSIS 3.0 IPv6-ready specification for cable modems in August 2006. The widely used DOCSIS 2.0 does not support IPv6. The new 'DOCSIS 2.0 + IPv6' standard also supports IPv6, which may on the cable modem side only require a firmware upgrade.[53] [54] It is expected that only 60% of cable modems' servers and 40% of cable modems will be DOCSIS 3.0 by 2011.[55]

Other equipment which is typically not IPv6-ready ranges from Voice over Internet Protocol devices to laboratory equipment and printers.

Deployment

The introduction of Classless Inter-Domain Routing (CIDR) in the Internet routing and IP address allocation methods in 1993 and the extensive use of network address translation (NAT) delayed the inevitable IPv4 address exhaustion, but the final phase of exhaustion started on February 3, 2011.[17] However, despite a decade long development and implementation history as a Standards Track protocol, general worldwide deployment is still in its infancy. As of October 2011, about 3% of domain names and 12% of the networks on the internet have IPv6 protocol support.[56]

Nevertheless, IPv6 has been implemented on all major operating systems in use in commercial, business, and home consumer environments.[57] Since 2008, the domain name system can be used in IPv6 as major web sites like Google, although sometimes with extra configuration[58] . IPv6 was first used in a major world event during the 2008 Summer Olympic Games[59] , the largest showcase of IPv6 technology since the inception of IPv6.[60] Countries like China or the Federal U.S. Government are also starting to require support for IPv6 on their equipment. Finally, modern cellular telephone specifications mandate IPv6 operation and deprecate IPv4 as an optional capability.[61]

Controversy

Privacy extensions are, except for the Windows platform and Mac OS X since 10.7 as well as iOS since version 4.3, not enabled by default. The fact that the unique MAC address is exposed to the Internet and therefore makes devices trackable has led to criticism in various countries. Two actions are necessary to guarantee the same level as with today's IPv4 networks: the client device has the privacy extensions enabled, and the provider dynamically assigns a varying address block to the client device.[62] [63] [64] [65] [66]

Notes

[1] RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden (December 1998)

[2] eweek.com (http://www.eweek.com/c/a/IT-Infrastructure/IPv4-Address-Exhaustion-Not-Instant-Cause-for-Concern-with-IPv6-in-Wings-287643/)

[3] Google: more Macs mean higher IPv6 usage in US (http://arstechnica.com/news.ars/post/20081113-google-more-macs-mean-higher-ipv6-usage-in-us.html)

[4] *Global IPv6 Statistics – Measuring the current state of IPv6 for ordinary users*, S. H. Gunderson (Google), RIPE 57 (Dubai, Oct 2008) (http://www.ripe.net/ripe/meetings/ripe-57/presentations/Colitti-Global_IPv6_statistics_-_Measuring_the_current_state_of_IPv6_for_ordinary_users_.7gzD.pdf)

[5] *Google IPv6 Conference 2008: What will the IPv6 Internet look like?* (http://www.youtube.com/watch?v=mZo69JQoLb8). Event occurs at 13:35. .

[6] RFC 1752 *The Recommendation for the IP Next Generation Protocol*, S. Bradner, A. Mankin, January 1995.

[7] RFC 1550, *IP: Next Generation (IPng) White Paper Solicitation*, S. Bradner, A. Mankin (December 1993)

[8] History of the IPng Effort (http://playground.sun.com/ipv6/doc/history.html)

[9] The working-group members were J. Allard (Microsoft), Steve Bellovin (AT&T), Jim Bound (Digital Equipment Corporation), Ross Callon (Wellfleet), Brian Carpenter (CERN), Dave Clark (MIT), John Curran (NEARNET), Steve Deering (Xerox), Dino Farinacci (Cisco), Paul Francis (NTT), Eric Fleischmann (Boeing), Mark Knopper (Ameritech), Greg Minshall (Novell), Rob Ullmann (Lotus), and Lixia Zhang (Xerox).

[10] "River of IPv4 addresses officially runs dry" (http://arstechnica.com/tech-policy/news/2011/02/river-of-ipv4-addresses-officially-runs-dry). arsttechnica.com. .

[11] Rashid, Fahmida Y. (February 3, 2011). "IPv4 Address Depletion Adds Momentum to IPv6 Transition" (http://www.eweek.com/c/a/IT-Infrastructure/IPv4-Address-Depletion-Adds-Momentum-to-IPv6-Transition-875751/). *eWeek.com*. . Retrieved February 3, 2011.

[12] "Two /8s allocated to APNIC from IANA" (http://www.apnic.net/publications/news/2011/delegation). APNIC. 2010-01-01. . Retrieved 2011-02-03.

[13] Asia-Pacific Network Information Centre (15 April 2011). "APNIC IPv4 Address Pool Reaches Final /8" (http://www.apnic.net/publications/news/2011/final-8). . Retrieved 15 April 2011.

[14] Exec: No shortage of Net addresses (http://news.zdnet.com/2100-1009_22-1020653.html) By John Lui, CNETAsia

[15] A Pragmatic Report on IPv4 Address Space Consumption (http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-3/ipv4.html) by Tony Hain, Cisco Systems

[16] Proposed Global Policy for the Allocation of the Remaining IPv4 Address Space (http://www.ripe.net/ripe/policies/proposals/2008-03.html)

[17] IPv4 Address Report (http://www.potaroo.net/tools/ipv4/)
[18] RFC 1726, *Technical Criteria for Choosing IP The Next Generation (IPng)*, Partridge C., Kastenholz F. (December 1994)
[19] U.S. Census Bureau (http://www.census.gov/main/www/popclock.html)
[20] "Moving to IPv6: Now for the hard part (FAQ) | Deep Tech – CNET News" (http://news.cnet.com/8301-30685_3-20030482-264.html). . Retrieved 2011-02-03.
[21] RFC 2071, *Network Renumbering Overview: Why would I want it and what is it anyway?*, P. Ferguson, H. Berkowitz (January 1997)
[22] RFC 2072, *Router Renumbering Guide*, H. Berkowitz (January 1997)
[23] RFC 1112, *Host extensions for IP multicasting*, S. Deering (August 1989)
[24] RFC 3956, *Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address*, P. Savola, B. Haberman (November 2004)
[25] RFC 2908, *The Internet Multicast Address Allocation Architecture*, D. Thaler, M. Handley, D. Estrin (September 2000)
[26] RFC 3306
[27] RFC 4862, *IPv6 Stateless Address Autoconfiguration*, S. Thomson, T. Narten, T. Jinmei (September 2007)
[28] RFC 2894, *Router Renumbering for IPv6*, M. Crawford, August 2000.
[29] J. Loughney, Ed. (April 2006). "IPv6 Node Requirements" (http://tools.ietf.org/html/rfc4294#section-8). . Retrieved 7 December 2011. "Security Architecture for the Internet Protocol [RFC-4301] MUST be supported."
[30] UDP/IPv4 may actually have a checksum of 0, indicating no checksum; IPv6 requires UDP to have its own checksum.
[31] This improvement may have been made less necessary by the development of routers that perform checksum computation at link speed using dedicated hardware, but it is still relevant for software based routers.
[32] RFC 3963, *Network Mobility (NEMO) Basic Protocol Support*, V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert (January 2005)
[33] RFC 2675, *IPv6 Jumbograms*, D. Borman, S. Deering, R. Hinden (August 1999)
[34] RFC 4291 *IP Version 6 Addressing Architecture*, R. Hinden, S. Deering (February 2006)
[35] The sheer size of IPv6 (http://pthree.org/2009/03/08/the-sheer-size-of-ipv6/)
[36] RFC 4291 p. 9
[37] "IPv6 Address Allocation and Assignment Policy" (http://www.ripe.net/ripe/docs/ripe-512). RIPE NCC. 8 February 2011. . Retrieved 27 March 2011.
[38] "Comcast Activates First Users With IPv6 Native Dual Stack Over DOCSIS" (http://blog.comcast.com/2011/01/comcast-activates-first-users-with-ipv6-native-dual-stack-over-docsis.html). Comcast. 31 January 2011. .
[39] To convert a 48-bit MAC address to IPv6 EUI-64 format host address, first the 7th bit (scope bit) is flipped (so, for instance, 00 becomes 02 and vice-versa). Then 2 bytes with the value 0xFFFE are inserted between the vendor portion (the first 3 bytes) and the host portion (the last 3 bytes) of the MAC address. This pads the 48-bit address to 64 bits. "Recipe 25.1. Automatically Generating IPv6 Addresses for an Interface" (http://fengnet.com/book/Cisco.IOS.Cookbook.2nd/I_0596527225_CHP_25_SECT_2.html). . Retrieved 14 March 2011.
[40] IPv6 Transition Mechanism / Tunneling Comparison (http://www.sixxs.net/faq/connectivity/?faq=comparison)
[41] RFC4291 (http://tools.ietf.org/html/rfc4291)
[42] OpenBSD inet6(4) manual page (http://www.openbsd.org/cgi-bin/man.cgi?query=inet6&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html#PROTOCOLS)
[43] RFC 3493 - Basic Socket Interface Extensions for IPv6 (http://tools.ietf.org/html/rfc3493#page-22)
[44] RFC 3056 *Connection of IPv6 Domains via IPv4 Clouds*, B. Carpenter, Februari 2001.
[45] RFC 4380 *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*, C. Huitema, Februari 2006
[46] The Windows Vista Developer Story: Application Compatibility Cookbook (http://msdn2.microsoft.com/en-us/library/aa480152.aspx)
[47] RFC 5214 *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*, F. Templin, T. Gleeson, D. Thaler, March 2008.
[48] RFC 3053, *IPv6 Tunnel Broker*, A. Durand, P. Fasano, I. Guardini, D. Lento (January 2001)
[49] RFC 4966 Reasons to Move the Network Address Translator – Protocol Translator (NAT-PT) to Historic Status
[50] *Web sites must support IPv6 by 2012, expert warns* (http://www.networkworld.com/news/2010/012110-ipv6-warning.html). Network World. 21 January 2010. . Retrieved 2010-09-30
[51] RFC 5211
[52] "Networking IPv6 User Guide for JDK/JRE 5.0" (http://java.sun.com/j2se/1.5.0/docs/guide/net/ipv6_guide/index.html). . Retrieved 2007-09-30.
[53] "DOCSIS 2.0 Interface" (http://www.cablemodem.com/specifications/specifications20.html). Cablemodem.com. 2007-10-29. . Retrieved 2009-08-31.
[54] RMV6TF.org (http://rmv6tf.org/2008-IPv6-Summit-Presentations/Dan Torbet - IPv6andCablev2.pdf)
[55] "DOCSIS 3.0 Network Equipment Penetration to Reach 60% by 2011" (http://www.abiresearch.com/abiprdisplay.jsp?pressid=710) (Press release). ABI Research. 2007-08-23. . Retrieved 2007-09-30.
[56] Mike Leber (2010-10-02). "Global IPv6 Deployment Progress Report" (http://bgp.he.net/ipv6-progress-report.cgi). Hurricane Electric. . Retrieved 2011-10-19.
[57] Comparison of IPv6 support in operating systems Comparison of IPv6 support in operating systems
[58] http://www.google.com/intl/en/ipv6/
[59] "Beijing2008.cn leaps to next-generation Net" (http://en.beijing2008.cn/news/official/preparation/n214384681.shtml) (Press release). The Beijing Organizing Committee for the Games of the XXIX Olympiad. 2008-05-30. .

[60] Das, Kaushik (2008). "IPv6 and the 2008 Beijing Olympics" (http://ipv6.com/articles/general/IPv6-Olympics-2008.htm). *IPv6.com*. . Retrieved 2008-08-15. "As thousands of engineers, technologists have worked for a significant time to perfect this (IPv6) technology, there is no doubt, this technology brings considerable promises but this is for the first time that it will showcase its strength when in use for such a mega-event."

[61] Derek Morr (2009-06-09). "Verizon Mandates IPv6 Support for Next-Gen Cell Phones" (http://www.circleid.com/posts/20090609_verizon_mandates_ipv6_support_for_next_gen_cell_phones/). CircleID. .

[62] T. Narten, R. Draves (2001-01). "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" (http://www.ietf.org/rfc/rfc3041.txt). .

[63] Statement on IPv6 Address Privacy (ftp://ftp.cuhk.edu.hk/pub/doc/ipng/html/ipv6-address-privacy.html), Steve Deering & Bob Hinden, Co-Chairs of the IETF's IP Next Generation Working Group , 1999-11-06.

[64] IPv6: Privacy Extensions einschalten (http://www.heise.de/netze/artikel/IPv6-Privacy-Extensions-einschalten-1204783.html), Reiko Kaps, 2011-04-13

[65] Privacy Extensions (IPv6) (http://www.elektronik-kompendium.de/sites/net/1601271.htm), Elektronik Kompendium.

[66] Neues Internet-Protokoll erschwert anonymes Surfen (http://www.spiegel.de/netzwelt/web/0,1518,729340,00.html), Konrad Lischka, Spiegel Online, 2010-11-18.

References

External links

- IPv6 (http://www.dmoz.org/Computers/Internet/Protocols/IP/IPv6/) at the Open Directory Project
- IPv6 News – Daily Updated and IPv6 Portal + IPv6 Task Forces (http://www.ipv6tf.org/)
- IPv6 Backbone Network Topology (http://ipv6.nlsde.buaa.edu.cn)
- IPv4 Exhaustion Counter (http://www.IPv4ExhaustionCounter.com)
- Getipv6.info (http://www.getipv6.info) Technical information on IPv6 deployment from ARIN
- 6DEPLOY (http://www.6deploy.eu/) – IPv6 deployment support action within the European 7th Framework Programme, featuring an extensive list of tutorials (http://www.6deploy.eu/index.php?page=tutorials)
- List of product, services and applications with IPv6 support (http://www.ipv6-to-standard.org)
- Why IPv6 matters to radio stations (http://radioworld.com/article/why-ipv6-matters-to-your-station/23533)
- Security implications of implementing IPv6 (http://cert.inteco.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_security_implications_ipv6.pdf)
- Free Pool of IPv4 Address Space Depleted (http://www.nro.net/news/ipv4-free-pool-depleted)

IPv6 packet

An **IPv6 packet** is the smallest message entity exchanged via the Internet Protocol across an Internet Protocol version 6 (IPv6) network.

Packets consist of control information for addressing and routing, and a payload consisting of user data. The control information in IPv6 packets is subdivided into a mandatory fixed header and optional extension headers. The payload of an IPv6 packet is typically a datagram or segment of the higher-level Transport Layer protocol, but may be data for an Internet Layer (e.g., ICMPv6) or Link Layer (e.g., OSPF) instead.

IPv6 packets are typically transmitted over a Link Layer protocol, such as Ethernet which encapsulates each packet in a frame, but this may also be a higher layer tunneling protocol, such as IPv4 when using 6to4 or Teredo transition technologies.

Routers do not fragment IPv6 packets, as they do for IPv4. Hosts are "strongly recommended"[1] to implement path MTU discovery to take advantage of MTUs greater than the smallest MTU of 1280 octets. Hosts may use fragmentation to send packets larger than the observed path MTU.

Fixed header

The fixed header of an IPv6 packet consists of its first 40 octets (320 bits).[1] It has the following format:

Fixed header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*Traffic Class*								*Flow Label*																			
4	32	*Payload Length*																*Next Header*								*Hop Limit*							
8	64	*Source Address*																															
12	96																																
16	128																																
20	160																																
24	192	*Destination Address*																															
28	224																																
32	256																																
36	288																																

Version (4 bits)

The constant 6 (bit sequence `0110`).

Traffic Class (8 bits)

The bits of this field hold two values. The 6 most-significant bits are used for DSCP, which is used to classify packets.[2] [3] The remaining two bits are used for ECN;[4] priority values subdivide into ranges: traffic where the source provides congestion control and non-congestion control traffic.

Flow Label (20 bits)

Originally created for giving real-time applications special service.[1] Flow Label specifications and minimum requirements are described,[5] [6] and first uses of this field are emerging.[7]

Payload Length (16 bits)

The size of the payload in octets, including any extension headers. The length is set to zero when a *Hop-by-Hop* extension header carries a Jumbo Payload option.[8]

Next Header (8 bits)

Specifies the type of the next header. This field usually specifies the transport layer protocol used by a packet's payload. When extension headers are present in the packet this field indicates which extension header follows. The values are shared with those used for the IPv4 protocol field, as both fields have the same function (see List of IP protocol numbers).

Hop Limit (8 bits)

Replaces the time to live field of IPv4. This value is decremented by one at each intermediate node the packet visits. When the counter reaches 0 the packet is discarded.

Source Address (128 bits)

The IPv6 address of the sending node.

Destination Address (128 bits)

The IPv6 address of the destination node(s).

In order to increase performance, and since current link layer technology is assumed to provide sufficient error detection[9] , the header has no checksum to protect it.[1]

Extension headers

Extension headers carry optional Internet Layer information, and are placed between the fixed header and the upper-layer protocol header.[1] The headers form a chain, using the *Next Header* fields. The *Next Header* field in the fixed header indicates the type of the first extension header; the *Next Header* field of the last extension header indicates the type of the upper-layer protocol header in the payload of the packet.

All extension headers are a multiple of 8 octets in size; some extension headers require internal padding to meet this requirement.

There are several extension headers defined,[1] and new extension headers may be defined in the future. Extension headers are to be examined and processed at the packet's destination only, except for *Hop-by-Hop Options*, which need to be processed at every intermediate node on the packet's path, including sending and receiving node. The defined extension headers below are listed in the preferred order, should there be more than one extension header following the fixed header. Note that all extension headers are optional and should only appear at most once, except for the *Destination Options* header, which may appear twice.

If a node does not recognize a specific extension header, it should discard the packet and send an *Parameter Problem* message (ICMPv6 type 4, code 1).[1] When a Next Header value `0` appears in a header other than the fixed header a node should do the same.

Extension Header	Type	Description
Hop-by-Hop Options	0	Options that need to be examined by all devices on the path.
Destination Options (before routing header)	60	Options that need to be examined only by the destination of the packet.
Routing	43	Methods to specify the route for a datagram (used with Mobile IPv6).
Fragment	44	Contains parameters for fragmentation of datagrams.
Authentication Header (AH)	51	Contains information used to verify the authenticity of most parts of the packet.
Encapsulating Security Payload (ESP)	50	Carries encrypted data for secure communication.
Destination Options (before upper-layer header)	60	Options that need to be examined only by the destination of the packet.

Value 59 (No Next Header) in the Next Header field indicates that there is no next header *whatsoever* following this one, not even a header of an upper-layer protocol. It means that, from the header's point of view, the IPv6 packet ends right after it: the payload should be empty.[1] There could, however, still be data in the payload if the payload length in the first header of the packet is greater than the length of all extension headers in the packet. This data should be ignored by hosts, but passed unaltered by routers.

Hop-by-hop options and destination options

The *Hop-by-Hop Options* extension header needs to be examined by all nodes on the packet's path, including sending and receiving nodes. The *Destination Options* extension header need to be examined by the destination node(s) only. The extension headers are both at least 8 octets in size; if more options are present than will fit in that space, blocks of 8 octets are added to the header repeatedly—containing options and padding—until all options are represented.

Hop-by-Hop Options and *Destination Options* extension header format

Offsets	**Octet**	**0**								**1**								**2**								**3**							
Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	*Next Header*								*Hdr Ext Len*								*Options and Padding*															
4	**32**	*Options and Padding*																															
8	**64**	Optional: more *Options and Padding ...*																															
12	**96**																																

Next Header (8 bits)

Specifies the type of the next header.

Hdr Ext Len (8 bits)

Length of this header in 8-octet units, not including the first 8 octets.

Options (variable)

Contains one or more options, and optional padding fields to align options and to make the total header length a multiple of 8 octets. Options are TLV-coded.

Routing

The *Routing* extension header is used to direct a packet to one or more intermediate nodes before being sent to its destination. The header is at least 8 octets in size; if more *Type-specific Data* is needed than will fit in 4 octets, blocks of 8 octets are added to the header repeatedly, until all *Type-specific Data* is placed.[1]

Routing extension header format

Offsets	**Octet**	**0**								**1**								**2**								**3**							
Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	*Next Header*								*Hdr Ext Len*								*Routing Type*								*Segments Left*							
4	**32**	*Type-specific Data*																															
8	**64**	Optional: more *Type-specific Data* ...																															
12	**96**																																

Next Header (8 bits)

Indicates the type of the next header.

Hdr Ext Len (8 bits)

The length of this header, in multiples of 8 octets, not including the first 8 octets.

Routing Type (8 bits)

`0`, `1`, or `2`.

Segments Left (8 bits)

Number of nodes this packet still has to visit before reaching its final destination.

Type-specific Data (variable)

Data that belongs to this type of routing header.

Routing types

Due to the fact that with Routing Header type 0 a simple but effective[10] denial-of-service attack could be launched, this header is deprecated[11] and host and routers are required to ignore these headers.

Routing Header type 1 is used for the Nimrod[12] project funded by DARPA.

Routing Header type 2 is a limited version of type 0 and is used for Mobile IPv6, where it can hold the Home Address of the Mobile Node.

Fragment

In order to send a packet that is larger than the path MTU, the sending node splits the packet into fragments. The *Fragment* extension header carries the information necessary to reassemble the original (unfragmented) packet.[1]

Fragment extension header format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next Header*								*Reserved*								*Fragment Offset*													*Res*		*M*
4	32	*Identification*																															

Next Header (8 bits)

Identifies the type of the next header.

Reserved (8 bits)

Initialized to all zeroes.

Fragment Offset (13 bits)

Offset, in 8-octet units, relative to the start of the fragmentable part of the original packet.

Res (2 bits)

Reserved; initialized to zeroes.

M Flag (1 bit)

1 means more fragments follow; 0 means last fragment.

Identification (32 bits)

Packet identification value, generated by the source node. Needed for reassembly of the original packet.

Authentication Header (AH) and Encapsulating Security Payload (ESP)

The *Authentication Header* and the *Encapsulating Security Payload* are part of IPsec and are used identically in IPv6 and in IPv4.[13] [14]

Payload

The fixed and optional IPv6 headers are followed with the *upper-layer payload*, the data provided by the transport layer, for example a TCP segment or a UDP datagram. The *Next Header* field of the last IPv6 header indicates what type of payload is contained in this packet.

Standard payload length

The payload length field of IPv6 (and IPv4) has a size of 16 bits, capable of specifying a maximum size of 65535 octets for the payload. Most Link Layer protocols cannot process packets larger than 65535 octets.

Jumbogram

An optional feature of IPv6, the *jumbo payload* option in a *Hop-By-Hop Options* extension header[8] , allows the exchange of packets with payloads of up to one byte less than 4 GB ($2^{32} - 1 = 4294967295$ bytes), by making use of a 32-bit length field. Packets with such payloads are called jumbograms.

Since both TCP and UDP include fields limited to 16 bits (length, urgent data pointer), support for IPv6 jumbograms requires modifications to the Transport Layer protocol implementation.[8] Jumbograms are only relevant for links that have a MTU larger than 65583 octets (more than 65535 octets for the payload, plus 40 octets for the fixed header, plus 8 octets for the *Hop-by-Hop* extension header).

Fragmentation

Unlike in IPv4, IPv6 routers never fragment IPv6 packets. Packets exceeding the size of the maximum transmission unit of the destination link are dropped and this condition is signaled by a *Packet too Big* ICMPv6 type 2 message to the originating node, similarly to the IPv4 method when the *Don't Fragment* bit set.[1]

End nodes in IPv6 are expected to perform path MTU discovery to determine the maximum size of packets to send, and the upper-layer protocol is expected to limit the payload size. However, if the upper-layer protocol is unable to do so, the sending host may use the *Fragment* extension header in order to perform end-to-end fragmentation of IPv6 packets. Any data link layer conveying IPv6 data must be capable of delivering an IP packet containing 1280 bytes without the need to invoke end-to-end fragmentation at the IP layer.

Fragmenting

A packet containing a fragment of an original (larger) packet consists of two parts: the unfragmentable part of the original packet (which is the same for all fragments), and a piece of the fragmentable part of the original packet, identified by a fragment offset.

The unfragmentable part of a packet consists of the fixed header and some of the extension headers of the original packet (if present): all extension headers up to and including the *Routing* extension header, or else the *Hop-by-Hop* extension header. If neither extension headers are present, the unfragmentable part is just the fixed header.

The *Next Header* value of the last (extension) header of the unfragmentable part is set to `44` to indicate that a *Fragment* extension header follows. After the *Fragment* extension header a fragment of the rest of the original packet follows.

The first fragment(s) hold the rest of the extension headers (if present). After that the rest of the payload follows. Each fragment is a multiple of 8 octets in length, except the last fragment.

Each *Fragment* extension header has its *M* flag set to `1` (indicating more fragments follow), except the last, whose flag is set to `0`.

Reassembly

The original packet is reassembled by the receiving node by collecting all fragments and placing each fragment at the right offset and discarding the *Fragment* extension headers of the packets that carried them. Packets containing fragments need not arrive in sequence; they will be rearranged by the receiving node.

If not all fragments are received within 60 seconds after receiving the first packet with a fragment, reassembly of the original packet is abandoned and all fragments are discarded. If the *first* fragment was received (which contains the fixed header), a *Time Exceeded* message (ICMPv6 type 3, code 1) is returned to the node originating the fragmented packet, if the packet was discarded for this reason.

Receiving hosts must make a best-effort attempt to reassemble fragmented IP datagrams that, after reassembly, contain up to 1500 bytes. Hosts are permitted to make an attempt to reassemble fragmented datagrams larger than 1500 bytes, but they are also permitted to silently discard any datagram after it becomes apparent that the reassembled packet would be larger than 1500 bytes. Therefore, senders should avoid sending fragmented IP datagrams with a total reassembled size larger than 1500 bytes, unless they have previous assurance that the receiver is capable of reassembling such large datagrams.

References

[1] Deering, S.; Hinden, R. (December 1998). Internet Protocol, version 6 (IPv6) Specification (http://tools.ietf.org/html/rfc2460). IETF. RFC 2460.

[2] Nickols, K.; Blake, S.; Baker, F.; Black, D. (December 1998) *Definition of the Differentiated Service Field (DS Field) in the IPv4 and IPv6 Headers* (http://tools.ietf.org/html/rfc2474.html), IETF. RFC 2474.

[3] Grossman, D. (April 2002) *New Terminology and Clarifications for DiffServ* (http://tools.ietf.org/html/rfc3260.html), IETF. RFC 3260.

[4] Ramakrishnan, K.; Floyd, S.; Black, D. (September 2001) *The Addition of Explicit Congestion Notification (ECN) to IP* (http://tools.ietf.org/html/rfc3168.html), IETF. RFC 3168.

[5] Wijnen, B. (September 2003) *Textual Conventions for IPv6 Flow Label* (http://tools.ietf.org/html/rfc3595), IETF. RFC 3595.

[6] Rajahalme, J.; Conta, A.; Carpenter, B.; Deering, S. (March 2004) *IPv6 Flow Label Specification* (http://tools.ietf.org/html/rfc3697), IETF. RFC 3697.

[7] draft-blake-ipv6-flow-label-nonce-02 (http://tools.ietf.org/html/draft-blake-ipv6-flow-label-nonce-02)

[8] Borman, D.; Deering, S.; Hinden, R. (August 1999). *IPv6 Jumbograms* (http://tools.ietf.org/html/rfc2675). IETF. RFC 2675.

[9] RFC 1726 section 6.2

[10] Philippe Biondi, Arnoud Ebalard (April 2007). "IPv6 Routing Header Security" (http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf) (pdf). EADS. . Retrieved 3 December 2010. "Type 0: the evil mechanism..."

[11] Abley, J.; Savola, P.; Neville-Neil, G. (December 2007). *Deprecation of Type 0 Routing Headers in IPv6* (http://tools.ietf.org/html/rfc5095). IETF. RFC 5095.

[12] Castineyra, I.; Chiappa, N.; Steenstrup, M. (Augustus 1996) *The Nimrod Routing Architecture* (http://tools.ietf.org/html/rfc1992)*', IETF. RFC 1992.*

[13] Kent, S. (December 2005) *IP Authentication Header* (http://tools.ietf.org/html/rfc4302.html) IETF. RFC 4202.

[14] Kent, S. (December 2005) *IP Encapsulating Security Payload* (http://tools.ietf.org/html/rfc4303.html) IETF. RFC 4203.

Maximum transmission unit

In computer networking, the **maximum transmission unit (MTU)** of a communications protocol of a layer is the size (in bytes) of the largest protocol data unit that the layer can pass onwards. MTU parameters usually appear in association with a communications interface (NIC, serial port, etc.). Standards (Ethernet, for example) can fix the size of an MTU; or systems (such as point-to-point serial links) may decide MTU at connect time.

A larger MTU brings greater efficiency because each packet carries more user data while protocol overheads, such as headers or underlying per-packet delays, remain fixed; the resulting higher efficiency means a slight improvement in bulk protocol throughput. A larger MTU also means processing of fewer packets for the same amount of data. In some systems, per-packet-processing can be a critical performance limitation.

Large packets can occupy a slow link for some time, causing greater delays to following packets and increasing lag and minimum latency. For example, a 1500-byte packet, the largest allowed by Ethernet at the network layer (and hence over most of the Internet), ties up a 14.4k modem for about one second.

Large packets are also problematic in the presence of communications errors. Corruption of a single bit in a packet requires that the entire packet be retransmitted. At a given bit error rate larger packets are more likely to be corrupted. Retransmissions of larger packets take longer.

Table of MTUs of common media

Note: the MTUs in this section are given as the maximum size of IP packet that can be transmitted without fragmentation - including IP headers but excluding headers from lower levels in the protocol stack. The MTU must not be confused with the minimum datagram size that all hosts must be prepared to accept, which has a value of 576 for IPv4[1] and of 1280 for IPv6.[2]

Media	Maximum Transmission Unit (bytes)	Notes
Internet IPv4 Path MTU	At least 68[3]	Practical path MTUs are generally higher. IPv4 links must be able to forward packets of size up to 68 bytes. Systems may use Path MTU Discovery[4] to find the actual path MTU. This should not be mistaken with the packet size every host must be able to handle, which is 576.[5]
Internet IPv6 Path MTU	At least 1280[6]	Practical path MTUs are generally higher. Systems must use Path MTU Discovery[7] to find the actual path MTU.
Ethernet v2	1500[4]	Nearly all IP over Ethernet implementations use the Ethernet V2 frame format.
Ethernet (802.3)	1492[4]	
Ethernet Jumbo Frames	1500-9000	The limit varies by vendor. For correct interoperation, the whole Ethernet network must have the same MTU. Jumbo frames are usually only seen in special purpose networks.
WLAN (802.11)	2272[8]	
Token Ring (802.5)	4464	
FDDI	4352[4]	

IP (Internet protocol)

DARPA designed the Internet protocol suite to work over many networking technologies, each of which may use packets of different size. While a host will know the MTU of its own interface and possibly that of its peers (from initial handshakes), it will not initially know the lowest MTU in a chain of links to any other peers. Another potential problem is that higher-level protocols may create packets larger than a particular link supports.

To get around this issue, IP allows fragmentation: dividing the datagram into pieces, each small enough to pass over the single link that is being fragmented for, using the MTU parameter configured for that interface. This fragmentation process takes place at the IP layer (OSI layer 3) and marks packets it fragments as such, so that the IP layer of the destination host knows it should reassemble the packets into the original datagram. This method implies a number of possible drawbacks:

- All fragments of a packet must arrive for the packet to be considered received. If the network drops any fragment, the entire packet is lost.
- When the size of most or all packets exceed the MTU of a particular link that has to carry those packets, almost everything has to be fragmented. In certain cases the overhead this causes can be considered unreasonable or unnecessary. For example, various tunneling situations cross the MTU by very little as they add just a header's worth of data. The addition is small, but each packet now has to be sent in two fragments, the second of which carries very little payload. The same amount of payload is being moved, but every intermediate router has to do double the work in terms of header parsing and routing decisions.
- As it is normal to maximize the payload in every fragment, in general as well as when fragmenting, any further fragmentation that turns out to be necessary will increase the overhead even more.
- There is no simple method to discover the MTU of links beyond a node's direct peers.

The Internet Protocol requires that hosts must be able to process IP datagrams of at least 576 bytes (for IPv4) or 1280 bytes (for IPv6). However, this does not preclude Data Link Layers with an MTU smaller than IP's minimum MTU from conveying IP data. For example, according to IPv6's specification, if a particular Data Link Layer physically cannot deliver an IP datagram of 1280 bytes in a single frame, then the link layer MUST provide its own fragmentation and reassembly mechanism, separate from IP's own fragmentation mechanism, to ensure that a 1280-byte IP datagram can be delivered, intact, to the IP layer.

Path MTU Discovery

The Internet Protocol defines the "Path MTU" of an Internet transmission path as the smallest MTU of any of the IP hops of the "path" between a source and destination. Put another way, the path MTU is the largest packet size that can traverse this path without suffering fragmentation.

RFC 1191 (IPv4) and RFC 1981 (IPv6) describe "Path MTU Discovery", a technique for determining the path MTU between two IP hosts. It works by setting the DF (Don't Fragment) option in the IP headers of outgoing packets. Any device along the path whose MTU is smaller than the packet will drop such packets and send back an ICMP "Destination Unreachable (Datagram Too Big)" message containing its MTU. This information allows the source host to reduce its assumed path MTU appropriately. The process repeats until the MTU becomes small enough to traverse the entire path without fragmentation.

Unfortunately, increasing numbers of networks drop ICMP traffic (e.g. to prevent denial-of-service attacks), which prevents path MTU discovery from working. One often detects such blocking in the cases where a connection works for low-volume data but hangs as soon as a host sends a large block of data. For example, with IRC a connecting client might see the initial messages up to and including the initial ping (sent by the server as an anti spoofing measure), but get no response after that. This is because the large set of welcome messages are sent out in packets bigger than the real MTU. Also, in an IP network, the path from the source address to the destination address often gets modified dynamically, in response to various events (load-balancing, congestion, outages, etc.) - this could result in the path MTU changing (sometimes repeatedly) during a transmission, which may introduce further packet

drops before the host finds the new safe MTU.

Most Ethernet LANs use an MTU of 1500 bytes (modern LANs can use Jumbo frames, allowing for an MTU up to 9000 bytes); however, border protocols like PPPoE will reduce this. The difference between the MTU seen by end-nodes (e.g. 1500) and the Path MTU causes Path MTU Discovery to come into effect, with the possible result of making some sites behind badly-configured firewalls unreachable. One can possibly work around this, depending on which part of the network one controls; for example one can change the MSS (maximum segment size) in the initial packet that sets up the TCP connection at one's firewall.

RFC 4821, Packetization Layer Path MTU Discovery, describes a Path MTU Discovery technique which responds more robustly to ICMP filtering.

ATM backbones, an example of MTU tuning

Sometimes the demands of efficiency encourage artificially declaring a reduced MTU in software below the true maximum possible length supported - for example: where an ATM (Asynchronous Transfer Mode) network carries IP traffic. Some providers, particularly those with a telephony background, use ATM on their internal backbone network.

ATM operates at optimum efficiency when packet length is a multiple of 48 bytes. This is because ATM is sent as a stream of fixed-length packets (known as 'cells'), each of which can carry a payload of 48 bytes of user data with 5 bytes of overhead for a total cost of 53 bytes per cell. So the total length of the transmitted data length is 53 * ncells bytes, where ncells = the number of required cells of = INT((payload_length+47)/48). So in the worst case, where the total length = (48*n+1) bytes, one additional cell is needed to transmit the one last byte of payload, the final cell costing an extra 53 transmitted bytes 47 of which are padding. For this reason, artificially declaring a reduced MTU in software maximises protocol efficiency at the ATM layer by making the ATM AAL5 total payload length a multiple of 48 bytes whenever possible.

For example, 31 completely filled ATM cells carry a payload of 31*48=1488 bytes. Taking this figure of 1488 and subtracting from it any overheads contributed by all relevant higher protocols we can obtain a suggested value for an artificially-reduced optimal MTU. In the case where the user would normally send 1500 byte packets, sending between 1489 and 1536 bytes requires an additional fixed cost of 53 bytes transmitted, in the form of one extra ATM cell.

For the example of IP over DSL connections using PPPoA/VC-MUX, again choosing to fill 31 ATM cells as before, we obtain a desired optimal reduced MTU figure of 1478 = 31*48-10 taking into account an overhead of 10 bytes consisting of a Point-to-Point Protocol overhead of 2 bytes, and an AAL5 overhead of 8 bytes. This gives a total cost of 31*53=1643 bytes transmitted via ATM from a 1478 byte packet passed to PPPoA. In the case of IP sent over ADSL using PPPoA the figure of 1478 would be the total length of the IP packet including IP headers. So in this example, keeping to a self-imposed reduced MTU of 1478 as opposed to sending IP packets of total length 1500 saves 53 bytes per packet at the ATM layer at a cost of a 22 byte reduction of the length of IP packets.

RFC 2516 prescribes a maximum MTU for PPPoE/DSL connections of 1492 bytes: the 1500 byte maximum ethernet payload minus 8 bytes of PPPoE headers (2 bytes for the PPP overhead, and 6 bytes for the PPPoE header). This will not necessarily fill an integer number of ATM cells.

MTU in other standards

The G.hn standard, developed by ITU-T, provides a high-speed (up to 1 Gigabit/s) local area network using existing home wiring (power lines, phone lines and coaxial cables). The G.hn Data Link Layer accepts data frames of up to 2^{14} bytes (16384 bytes). In order to avoid the problem of long data-frames taking up the medium for long periods of time, G.hn defines a procedure for segmentation that divides the data frame into smaller segments.

Disruption

The transmission of a packet on a physical network segment that is larger than the segment's MTU is known as *jabber*. This is almost always caused by faulty devices. Many network switches have a built-in capability to detect when a device is jabbering and block it until it resumes proper operation.[9]

References

[1] RFC 791, p. 13
[2] RFC 2460, p. 13
[3] RFC 791, p. 24, "Every internet module must be able to forward a datagram of 68 octets without further fragmentation."
[4] RFC 1191
[5] RFC 791, p. 24, "Every internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled."
[6] RFC 2460
[7] RFC 6145
[8] Structure of the IEEE 802.11 MAC Frames - Wireless,Wlan,wifi,Configuration,and,Optimization Tips (http://www.wireless-center.net/Wireless-Internet-Technologies-and-Applications/1925.html)
[9] 3Com SuperStack Switch Management Guide (http://support.3com.com/infodeli/tools/switches/ss3/management/ug/cli_mg6a.htm)

- Marc Slemko (January 18, 1998). "Path MTU Discovery and Filtering ICMP" (http://alive.znep.com/~marcs/mtu/index.html). Retrieved 2007-09-02.

External links

- Tweaking your MTU / RWin for Orange Broadband Users (http://www.orangeproblems.co.uk/kitz/)
- How to set the TCP MSS value using iptables (http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html#TCPMSSTARGET)
- Discovering of MTU value (http://help.expedient.net/broadband/mtu_ping_test.shtml) via ping and setting it in Microsoft Windows
- DrTCP (http://www.dslreports.com/drtcp) – a utility for optimizing MTU under Microsoft Windows
- mturoute (http://www.elifulkerson.com/projects/mturoute.php) – a console utility for debugging mtu problems
- MSS Initiative (http://www.phildev.net/mss/)
- MTU Path (http://www.iea-software.com/products/mtupath.cfm) MTU discovery tool for IPv4 and IPv6 networks

Mobile IP

Mobile IP (or **IP mobility**) is an Internet Engineering Task Force (IETF) standard communications protocol that is designed to allow mobile device users to move from one network to another while maintaining a permanent IP address. Mobile IP for IPv4 is described in IETF RFC 5944, and extensions are defined in IETF RFC 4721. **Mobile IPv6**, the IP mobility implementation for the next generation of the Internet Protocol, IPv6, is described in RFC 6275.

Introduction

The Mobile IP protocol allows location-independent routing of IP datagrams on the Internet. Each mobile node is identified by its home address disregarding its current location in the Internet. While away from its home network, a mobile node is associated with a *care-of* address which identifies its current location and its home address is associated with the local endpoint of a tunnel to its *home agent*. Mobile IP specifies how a mobile node registers with its home agent and how the home agent routes datagrams to the mobile node through the *tunnel*.

Applications

In many applications (e.g., VPN, VoIP), sudden changes in network connectivity and IP address can cause problems. Mobile IP was designed to support seamless and continuous Internet connectivity.

Mobile IP is most often found in wired and wireless environments where users need to carry their mobile devices across multiple LAN subnets. Examples of use are in roaming between overlapping wireless systems, e.g., IP over DVB, WLAN, WiMAX and BWA.

Mobile IP is not required within cellular systems such as 3G, to provide transparency when Internet users migrate between cellular towers, since these systems provide their own data link layer handover and roaming mechanisms. However, it is often used in 3G systems to allow seamless IP mobility between different packet data serving node (PDSN) domains.

Operational principles

A mobile node has two addresses - a permanent home address and a care-of address (CoA), which is associated with the network the mobile node is visiting. Two kinds of entities comprise a Mobile IP implementation:

- A *home agent* stores information about mobile nodes whose permanent home address is in the home agent's network.
- A *foreign agent* stores information about mobile nodes visiting its network. Foreign agents also advertise care-of addresses, which are used by Mobile IP. If there is no foreign agent in the host network, the mobile device has to take care of getting an address and advertising that address by its own means.

A node wanting to communicate with the mobile node uses the permanent home address of the mobile node as the destination address to send packets to. Because the home address logically belongs to the network associated with the home agent, normal IP routing mechanisms forward these packets to the home agent. Instead of forwarding these packets to a destination that is physically in the same network as the home agent, the home agent redirects these packets towards the remote address through an IP tunnel by encapsulating the datagram with a new IP header using the care of address of the mobile node.

When acting as transmitter, a mobile node sends packets directly to the other communicating node, without sending the packets through the home agent, using its permanent home address as the source address for the IP packets. This is known as triangular routing. If needed, the foreign agent could employ *reverse tunneling* by tunneling the mobile node's packets to the home agent, which in turn forwards them to the communicating node. This is needed in

networks whose gateway routers check that the source IP address of the mobile host belongs to their subnet or discard the packet otherwise.

Performance

A performance evaluation of Mobile IPv6 can be found in [1] Additionally, a performance comparison between Mobile IPv6 and some of its proposed enhancements (Hierarchical Mobile IPv6, Fast Handovers for Mobile IPv6 and their Combination) is available at [2] .

Development

Enhancements to the Mobile IP technique, such as Mobile IPv6[3] and Hierarchical Mobile IPv6 (HMIPv6) defined in RFC 5380 [4],[5] are being developed to improve mobile communications in certain circumstances by making the processes more secure and more efficient. HMIPv6 explanation can be found at Hierarchical-Mobile-IPv6 [6].

Researchers create support for mobile networking without requiring any pre-deployed infrastructure as it currently is required by MIP. One such example is Interactive Protocol for Mobile Networking (IPMN) [7] which promises supporting mobility on a regular IP network just from the network edges by intelligent signalling between IP at end-points and application layer module with improved quality of service.

Researchers are also working to create support for mobile networking between entire subnets with support from Mobile IPv6. One such example is Network Mobility (NEMO) Network Mobility Basic Support Protocol [8] by the IETF Network Mobility Working Group [9] which supports mobility for entire Mobile Networks that move and to attach to different points in the Internet. The protocol is an extension of Mobile IPv6 and allows session continuity for every node in the Mobile Network as the network moves.

Changes in IPv6 for Mobile IPv6

- A set of mobility options to include in mobility messages
- A new Home Address option for the Destination Options header
- A new Type 2 Routing header
- New Internet Control Message Protocol for IPv6 (ICMPv6) messages to discover the set of home agents and to obtain the prefix of the home link
- Changes to router discovery messages and options and additional Neighbor Discovery options

Definition of terms

Home network

The home network of a mobile device is the network within which the device receives its identifying IP address (home address).

Home address

The home address of a mobile device is the IP address assigned to the device within its home network.

Foreign network

A foreign network is the network in which a mobile node is operating when away from its home network.

Care-of address

The care-of address of a mobile device is the network-native IP address of the device when operating in a foreign network.

Home agent

A home agent is a router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home. It maintains current location (IP address) information for the mobile node. It is used with one or more foreign agents.

Foreign agent

A foreign agent is a router that stores information about mobile nodes visiting its network. Foreign agents also advertise care-of-addresses which are used by Mobile IP.

Binding

A binding is the association of the home address with a care-of address.

References

[1] X.Pérez-Costa and H.Hartenstein. A Simulation Study on the Performance of Mobile IPv6 in a WLAN-Based Cellular Network. (http://portal.acm.org/citation.cfm?id=604022.604035&coll=&dl=ACM). Elsevier Computer Networks Journal (CNJ), special issue on The New Internet Architecture, September 2002.

[2] X.Pérez-Costa, M.Torrent-Moreno and H.Hartenstein. A Performance Comparison of Mobile IPv6, Hierarchical Mobile IPv6, Fast Handovers for Mobile IPv6 and their Combination. (http://portal.acm.org/citation.cfm?id=965736) ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), Volume 7, Issue 4, October, 2003.

[3] X.Pérez-Costa and H.Hartenstein. A Simulation Study on the Performance of Mobile IPv6 in a WLAN-Based Cellular Network (http://portal.acm.org/citation.cfm?id=604022.604035&coll=&dl=ACM) Elsevier Computer Networks Journal, special issue on The New Internet Architecture, September 2002

[4] http://tools.ietf.org/html/rfc5380

[5] X.Pérez-Costa, M.Torrent-Moreno and H.Hartenstein. A Simulation Study on the Performance of Hierarchical Mobile IPv6 (http://dsn.tm.uka.de/medien/publication-confs/perez-itc03-simulation-study.pdf) In Proceedings of the International Teletraffic Congress (ITC), Berlin, Germany, August 2003.

[6] http://searchmobilecomputing.techtarget.com/definition/Hierarchical-Mobile-IPv6

[7] http://medianet.kent.edu/ipmn/main.html

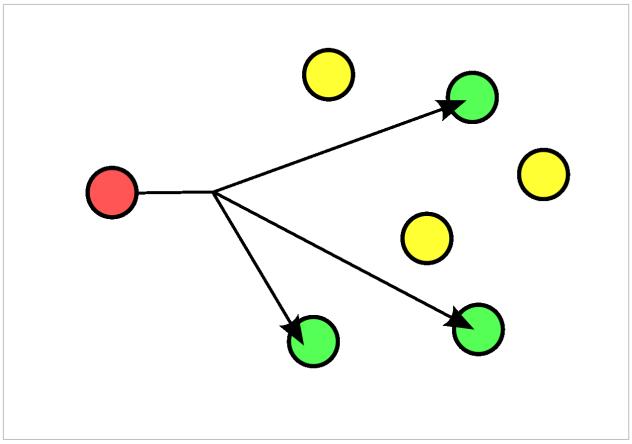
[8] http://www.rfc-editor.org/rfc/rfc3963.txt

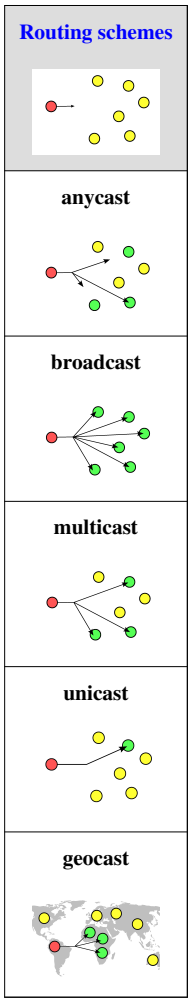
[9] http://tools.ietf.org/wg/nemo/

External links

- RFC 6275 -Mobility support for IPv6
- RFC 5944 - IP Mobility Support for IPv4, Revised
- RFC 4721 - Mobile IPv4 Challenge/Response Extensions
- RFC 3024 - Reverse Tunneling for Mobile IP
- Inside Mobile IP (http://www.ddj.com/dept/mobile/184406240)
- Protocols for Adaptive Mobile and Wireless Networking (http://www.monarch.cs.cmu.edu/)
- Mobile IP explained (a tutorial) (http://www.tml.tkk.fi/Opinnot/Tik-111.550/1999/Esitelmat/MobileIP/Mobip.html)
- Mobility Extensions for IPv6 (mext) IETF Working Group Web site (http://ietf.org/html.charters/mext-charter.html)
- Mobile IPv6 -- A short introduction (http://www.hznet.de/ipv6/mipv6-intro.pdf) by Holger Zuleger
- Linux Mobile IPv6 HOWTO (http://tldp.org/HOWTO/Mobile-IPv6-HOWTO/) on the Linux Documentation Project
- D. Johnson, C. Perkins, J. Arkko. Mobility Support in IPv6 (http://tools.ietf.org/html/rfc6275). RFC 6275. June 2011
- J. Arkko, V. Devarapalli, F. Dupont. Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents (http://www.ietf.org/rfc/rfc3776.txt). RFC 3776. June 2004

Multicast





In computer networking, **multicast** is the delivery of a message or information to a group of destination computers simultaneously in a single transmission from the source creating copies automatically in other network elements, such as routers, only when the topology of the network requires it.

Multicast is most commonly implemented in IP multicast, which is often employed in Internet Protocol (IP) applications of streaming media and Internet television. In IP multicast the implementation of the multicast concept occurs at the IP routing level, where routers create optimal distribution paths for datagrams sent to a multicast destination address.

At the Data Link Layer, *multicast* describes one-to-many distribution such as Ethernet multicast addressing, Asynchronous Transfer Mode (ATM) point-to-multipoint virtual circuits (P2MP) or Infiniband multicast.

IP multicast

IP multicast is a technique for one-to-many communication over an IP infrastructure in a network. It scales to a larger receiver population by not requiring prior knowledge of who or how many receivers there are. Multicast uses network infrastructure efficiently by requiring the source to send a packet only once, even if it needs to be delivered to a large number of receivers. The nodes in the network take care of replicating the packet to reach multiple receivers only when necessary.

The most common transport layer protocol to use multicast addressing is User Datagram Protocol (UDP). By its nature, UDP is not reliable—messages may be lost or delivered out of order. Reliable multicast protocols such as Pragmatic General Multicast (PGM) have been developed to add loss detection and retransmission on top of IP multicast.

IP multicast is widely deployed in enterprises, commercial stock exchanges, and multimedia content delivery networks. A common enterprise use of IP multicast is for IPTV applications such as distance learning and televised company meetings.

Other multicast technologies

As of 2006, most effort at scaling multicast up to large networks have concentrated on the simpler case of single-source multicast, which seems to be more computationally tractable.

Still, the large state requirements in routers make applications using a large number of trees unworkable using IP multicast. Take presence information as an example where each person needs to keep at least one tree of its subscribers, if not several. No mechanism has yet been demonstrated that would allow the IP multicast model to scale to millions of senders and millions of multicast groups and, thus, it is not yet possible to make fully general multicast applications practical. For these reasons, and also reasons of economics, IP multicast is not, in general, used in the commercial Internet backbone. The increasing availability of WiFi Access Points that support multicast IP is facilitating the emergence of WiCast WiFi Multicast that allows the binding of data to geographical locations.

Explicit Multi-Unicast (XCAST) is an alternate multicast strategy to IP multicast that provides reception addresses of all destinations with each packet. As such, since the IP packet size is limited in general, XCAST cannot be used for multicast groups of large number of destinations. The XCAST model generally assumes that the stations participating in the communication are known ahead of time, so that distribution trees can be generated and resources allocated by network elements in advance of actual data traffic.

Other multicast technologies not based on IP multicast are more widely used. Notably the Internet Relay Chat (IRC), which is more pragmatic and scales better for large numbers of small groups. IRC implements a single spanning tree across its overlay network for all conference groups. This leads to suboptimal routing for some of these groups however. Additionally, IRC keeps a large amount of distributed states that limit growth of an IRC network, leading to fractioning into several non-interconnected networks.[1] The lesser known PSYC technology uses custom multicast strategies per conference.[2] Also some peer-to-peer technologies employ the multicast concept when distributing content to multiple recipients.

References

[1] RFC 1324 2.5.1 Size
[2] PSYC white paper (http://www.psyc.eu/whitepaper/)

External links

- Illustrative video about IP multicast (http://cisco.com/en/US/products/ps6552/products_ios_technology_home.html) at Cisco.
- IETF MBONE Deployment (mboned) Working Group (http://datatracker.ietf.org/wg/mboned/charter)
- IETF Multicast & Anycast Group Membership (magma) Working Group (http://ietf.org/html.charters/magma-charter.html)
- IETF Multicast Mobility (multimob) Working Group (http://datatracker.ietf.org/wg/multimob/charter)
- IETF Multicast Security (msec) Working Group (http://ietf.org/html.charters/msec-charter.html)
- IETF Protocol Independent Multicast (pim) Working Group (http://ietf.org/html.charters/pim-charter.html)
- IETF Reliable Multicast Transport (rmt) Working Group (http://ietf.org/html.charters/rmt-charter.html) (Aug. 2008)
- IETF Source-Specific Multicast (ssm) Working Group (http://ietf.org/html.charters/OLD/ssm-charter.html)
- IP-Ethernet multicast (http://www.firewall.cx/multicast-intro.php) tutorial.
- IRTF Scalable Adaptive Multicast (SAM) Research Group (http://www.irtf.org/charter?gtype=rg&group=samrg)
- Multicast IP details at sockets.com (http://sockets.com/ch16.htm#Multicast)
- Multicast on Packet Radio (http://uspacket.org/)
- Multicast over TCP/IP HOWTO (http://tldp.org/HOWTO/Multicast-HOWTO.html) (Mar. 1998) describes Multicast in the Linux kernel, although some sections (specially multicast programs) are outdated and do not cover recent software.
- qpimd - PIM Daemon for Quagga (http://www.nongnu.org/qpimd/): Multicast PIM module for the Quagga Routing Suite
- RFC 1112: Host Extensions for IP Multicasting, incl. Mapping von IPv4-Multicast-Adressen auf MAC-Adressen
- RFC 3170: IP Multicast Applications: Challenges and Solutions
- RFC 3180: GLOP Addressing in 233/8
- RFC 4608: Source-Specific Protocol Independent Multicast in 232/8
- RFC 5757: Multicast Mobility in Mobile IP Version 6 (MIPv6): Problem Statement and Brief Survey
- draft-irtf-samrg-common-api (http://tools.ietf.org/html/draft-irtf-samrg-common-api) A Common API for Transparent Hybrid Multicast
- Source-specific multicasts for JAVA platform (https://jssm.dev.java.net/)
- Southampton University multicast commentary (http://www.multicast.org.uk)
- XCAST (http://xcast.jp), a commercial MediaWiki-powered website based in Japan (English-language), devoted to Explicit Multicast technology.
- Xorp (http://xorp.org), a free software multicast router
- AMT - Automatic IP Multicast without explicit Tunnels (EBU Technical Review article) (http://tech.ebu.ch/docs/techreview/trev_2010-Q4_AMT_Kernen_Simlo.pdf)

Multicast address

A **multicast address** is a logical identifier for a group of hosts in a computer network, that are available to process datagrams or frames intended to be multicast for a designated network service. Multicast addressing can be used in the Link Layer (Layer 2 in the OSI model), such as Ethernet multicast, and at the Internet Layer (Layer 3 for OSI) for Internet Protocol Version 4 (IPv4) or Version 6 (IPv6) multicast.

IPv4

IPv4 multicast addresses are defined by the leading address bits of *1110*, originating from the classful network design of the early Internet when this group of addresses was designated as *Class D*. The Classless Inter-Domain Routing (CIDR) prefix of this group is `224.0.0.0/4`. The group includes the addresses from 224.0.0.0 to 239.255.255.255. Address assignments from within this range are specified in RFC 5771, an Internet Engineering Task Force (IETF) *Best Current Practice* document (BCP 51).

The following table is a list of notable well-known IPv4 addresses that are reserved for IP multicasting and that are registered with the Internet Assigned Numbers Authority (IANA).[1]

IP multicast address	Description
224.0.0.0	Base address (reserved)
224.0.0.1	The *All Hosts* multicast group addresses all hosts on the same network segment.
224.0.0.2	The *All Routers* multicast group addresses all routers on the same network segment.
224.0.0.4	This address is used in the Distance Vector Multicast Routing Protocol (DVMRP) to address multicast routers.
224.0.0.5	The Open Shortest Path First (OSPF) *All OSPF Routers* address is used to send Hello packets to all OSPF routers on a network segment.
224.0.0.6	The OSPF *All D Routers* address is used to send OSPF routing information to designated routers on a network segment.
224.0.0.9	The Routing Information Protocol (RIP) version 2 group address is used to send routing information to all RIP2-aware routers on a network segment.
224.0.0.10	The Enhanced Interior Gateway Routing Protocol (EIGRP) group address is used to send routing information to all EIGRP routers on a network segment.
224.0.0.13	Protocol Independent Multicast (PIM) Version 2
224.0.0.18	Virtual Router Redundancy Protocol (VRRP)
224.0.0.19 - 21	IS-IS over IP
224.0.0.22	Internet Group Management Protocol (IGMP) Version 3
224.0.0.102	Hot Standby Router Protocol version 2 (HSRPv2) / Gateway Load Balancing Protocol (GLBP)
224.0.0.107	Precision Time Protocol version 2 peer delay measurement messaging
224.0.0.251	Multicast DNS (mDNS) address
224.0.0.252	Link-local Multicast Name Resolution (LLMNR) address
224.0.1.1	Network Time Protocol clients listen on this address for protocol messages when operating in multicast mode.
224.0.1.39	The Cisco multicast router *AUTO-RP-ANNOUNCE* address is used by RP mapping agents to listen for candidate announcements.
224.0.1.40	The Cisco multicast router *AUTO-RP-DISCOVERY* address is the destination address for messages from the RP mapping agent to discover candidates.
224.0.1.41	H.323 Gatekeeper discovery address

224.0.1.129 - 132	Precision Time Protocol version 1 time announcements
224.0.1.129	Precision Time Protocol version 2 time announcements

Local subnetwork

Addresses in the range 224.0.0.0 to 224.0.0.255 are individually assigned by IANA and designated for multicasting on the local subnetwork only. For example, the Routing Information Protocol (RIPv2) uses 224.0.0.9, Open Shortest Path First (OSPF) uses 224.0.0.5 & 224.0.0.6, and Zeroconf mDNS uses 224.0.0.251. Routers must not forward these messages outside the subnet in which they originate.

Internetwork control block

Addresses in the range 224.0.1.0 to 224.0.1.255 are individually assigned by IANA and designated the Internetwork Control Block. It is used for traffic that must be routed through the public Internet, such as for applications of the Network Time Protocol (224.0.1.1).

AD-HOC block

Addresses in the ranges 224.0.2.0 to 224.0.255.255, 224.3.0.0 to 224.4.255.255 and 233.252.0.0 to 233.255.255.255 are individually assigned by IANA and designated the AD-HOC block. These addresses are globally routed and are used for applications that don't fit either of the previously described purposes.[2]

Source-specific multicast

The 232.0.0.0/8 block is reserved for use by source-specific multicast.

GLOP addressing

The 233.0.0.0/8 range was originally assigned by RFC 2770 as an experimental, public statically assigned multicast address space for publishers and Internet service providers that wished to source content on the Internet. The allocation method is termed GLOP addressing and provides implementers a block of 255 addresses that is determined by their 16-bit autonomous system number (ASN) allocation. In a nutshell, the middle two octets of this block are formed from assigned ASNs, giving any operator assigned an ASN 256 globally unique multicast group addresses. The method is not applicable to the newer 32-bit extension AS numbers. RFC 3180, superseding RFC 2770, envisioned the use of the range for many-to-many multicast applications. This block has been one of the most successful multicast addressing schemes. Unfortunately, with only 256 multicast addresses available to each autonomous system, GLOP is not adequate for large-scale broadcasters.[3]

Unicast-Prefix-Based IPv4 Multicast addresses

The 234.0.0.0/8 range is assigned by RFC 6034 as a range of global IPv4 multicast address space provided to each organization that has /24 or larger globally routed unicast address space allocated. A resulting advantage over GLOP is that the mechanisms in IPv4 and IPv6 become more similar.

Administratively scoped addresses

The 239.0.0.0/8 range is assigned by RFC 2365 as a locally administered address space with local or organizational scope. It may be used by anyone, without concern for address collisions, for private multicast domains.

The administratively scoped address block is further subdivided.

1. 239.255.0.0/16 is designated for local scope. Any scope boundary is a boundary for local scope. Local scope address space may be extended downwards if necessary (i.e. to 239.254.0.0/16, 239.253.0.0/16, etc.).[4]

2. 239.192.0.0/14 is designated for organizational local scope. This space may be extended to 239.0.0.0/10, 239.64.0.0/10 and 239.128.0.0/10 if necessary.[5]

Within each scope, the top 256 addresses are reserved for *relative assignments*. For example, the topmost relative assignment address is used by the Session Announcement Protocol.[6]

IPv6

Multicast addresses in IPv6 have the prefix `ff00::/8`. IPv6 multicast addresses are generally formed from four bit groups, illustrated as follows:

General multicast address format

Bits	8	4	4	112
Field	*prefix*	*flgs*	*sc*	*group ID*

The *prefix* holds the binary value `11111111` for any multicast address. Currently, 3 of the 4 flag bits in the *flgs* field are defined;[7] the most-significant flag bit is reserved for future use. The other three flags are known as *R*, *P* and *T*.

Multicast address flags[8]

Flag	0	1
R (Rendezvous)[9]	Rendezvous point not embedded	Rendezvous point embedded
P (Prefix)[10]	Without prefix information	Address based on network prefix
T (Transient)[11]	Well-known multicast address	Dynamically assigned multicast address

Similar to unicast addresses, the prefix of IPv6 multicast addresses specifies their scope, however, the set of possible scopes is different. The 4-bit *sc* (or scope) field (bits 12 to 15) is used to indicate where the address is valid and unique.

Multicast address scope

IPv6 address[12]	IPv4 equivalent[13]	Scope	Purpose
`ff00::/16-ff0f::/16`		Reserved	
`ffx1::/16`	`127.0.0.0/8`	Interface-local	Packets with this destination address may not be sent over any network link, but must remain within the current node; this is the multicast equivalent of the unicast loopback address.
`ffx2::/16`	`224.0.0.0/24`	Link-local	Packets with this destination address may not be routed anywhere.
`ffx3::/16`	`239.255.0.0/16`	IPv4 local scope	
`ffx4::/16`		Admin-local	The smallest scope that must be administratively configured.
`ffx5::/16`		Site-local	Restricted to the local physical network.

ffx8::/16	239.192.0.0/14	Organization-local	Restricted to networks used by the organization administering the local network. (For example, these addresses might be used over VPNs; when packets for this group are routed over the public internet (where these addresses are not valid), they would have to be encapsulated in some other protocol.)
ffxe::/16	224.0.1.0–238.255.255.255	Global scope	Eligible to be routed over the public internet.

The service is identified in the 112-bit *Group ID* field. For example, if `ff02::101` refers to all Network Time Protocol (NTP) servers on the local network segment, then `ff08::101` refers to all NTP servers in an organization's networks. The *Group ID* field may be further divided for special multicast address types.

The following table is a partial list of well-known IPv6 multicast addresses that are registered with the Internet Assigned Numbers Authority (IANA).[14]

Well-known IPv6 multicast addresses

Address	Description
ff02::1	All nodes on the local network segment
ff02::2	All routers on the local network segment
ff02::5	OSPFv3 AllSPF routers
ff02::6	OSPFv3 AllDR routers
ff02::9	RIP routers
ff02::a	EIGRP routers
ff02::d	PIM routers
ff02::16	MLDv2 reports (defined in RFC 3810)
ff02::1:2	All DHCP servers and relay agents on the local network site (defined in RFC 3315)
ff05::1:3	All DHCP servers on the local network site (defined in RFC 3315)
ff0x::fb	Multicast DNS
ff0x::101	Network Time Protocol
ff0x::108	Network Information Service
ff0x::114	Used for experiments

Ethernet

Ethernet frames with a value of 1 in the least-significant bit of the first octet[15] of the destination address are treated as multicast frames and are flooded to all points on the network. While frames with ones in all bits of the destination address (`FF:FF:FF:FF:FF:FF`) are sometimes referred to as broadcasts, Ethernet network equipment generally does not distinguish between multicast and broadcast frames. Modern Ethernet controllers filter received packets to reduce CPU load, by looking up the hash of a multicast destination address in a table, initialized by software, which controls whether a multicast packet is dropped or fully received.

Ethernet multicast address	Type Field	Usage
01-00-0C-CC-CC-CC	0x0802	CDP (Cisco Discovery Protocol), VTP (VLAN Trunking Protocol)
01-00-0C-CC-CC-CD	0x0802	Cisco Shared Spanning Tree Protocol Address
01-80-C2-00-00-00	0x0802	Spanning Tree Protocol (for bridges) IEEE 802.1D
01-80-C2-00-00-08	0x0802	Spanning Tree Protocol (for provider bridges) IEEE 802.1AD
01-80-C2-00-00-02	0x8809	Ethernet OAM Protocol IEEE 802.3ah
01-00-5E-xx-xx-xx	0x0800	IPv4 Multicast (RFC 1112)
33-33-xx-xx-xx-xx	0x86DD	IPv6 Multicast (RFC 2464)

|+Some well known Ethernet multicast addresses[16]

Notes

[1] IANA IP multicast addresses assignments (http://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt). Internet Assigned Numbers Authority .

[2] RFC 5771 Section 6.

[3] "Frequently Asked Questions (FAQ) File for Multicasting" (http://www.multicasttech.com/faq/). Multicast Tech. . Retrieved 2010-11-18.

[4] RFC 2365 section 6.1.

[5] RFC 2365 section 6.2.

[6] RFC 2365 section 9

[7] Hinden, R.; Deering, S. (February 2006) *IP Version 6 Addressing Architecture* (http://tools.ietf.org/html/rfc4291), IETF, RFC 4291.

[8] Silvia Hagen (May 2006). *IPv6 Essentials* (Second ed.). O'Reilly. ISBN 978-0596100582.

[9] RFC 3956

[10] RFC 3306

[11] RFC 4291

[12] *x* is a place holder indicating that the value of the *flags* field is unimportant in the current discussion.

[13] RFC 2365 section 8.

[14] IANA Internet Protocol Version 6 Multicast Addresses (http://www.iana.org/assignments/ipv6-multicast-addresses). Internet Assigned Numbers Authority.

[15] On Ethernet, the least-significant bit of an octet is the first to be transmitted. A multicast is indicated by the first transmitted bit of the destination address being 1.

[16] Patton, Michael A. et. al. (http://www.cavebear.com/archive/cavebear/Ethernet/acks.html). "Multicast (including Broadcast) Addresses" (http://www.cavebear.com/archive/cavebear/Ethernet/multicast.html). *cavebear.com*. Karl Auerbach. .

References

Multihoming

Multihoming is a technique used to increase the reliability of the Internet connection for an IP network. As an adjective, it is typically used to describe a customer, rather than an Internet service provider (ISP) network. In the context of competing business networks, platform multihoming refers to the condition of users affiliating with more than a single platform.

Multihoming variants

In the IP context, there are several ways to multihome, separate from the actual protocols used to do so, amongst which the most important are:

Single Link, Multiple IP address (spaces)

The host has multiple IP addresses (e.g. `2001:db8::1` and `2001:db8::2` in IPv6), but only one physical upstream link. When the single link fails, connectivity is down for all addresses.

Multiple Interfaces, Single IP address per interface

The host has multiple interfaces and each interface has one, or more, IP addresses. If one of the links fails, then its IP address becomes unreachable, but the other IP addresses will still work. Hosts that have multiple IPv6 or IPv4 records enabled can then still be reachable at the penalty of having the client program time out and retry on the broken address. Existing connections can't be taken over by the other interface, as TCP does not support this. To remedy this, one could use SCTP which does allow this situation. However SCTP is not used very much in practice.

Multiple Links, Single IP address (space)

This is what in general is meant with Multihoming. With the use of a routing protocol, in most cases BGP, the end-site announces this address space to its upstream links. When one of the links fails, the protocol notices this on both sides and traffic is not sent over the failing link any more. Usually this method is used to multihome a site and not for single hosts.

Multiple Links, Multiple IP address (spaces)

This approach uses a specialized Link Load Balancer (or WAN Load Balancer) appliance between the firewall and the link routers. No special configuration is required in the ISP's routers. It allows use of all links at the same time to increase the total available bandwidth and detects link saturation and failures in real time to redirect traffic. Algorithms allow traffic management. Incoming balancing is usually performed with a real time DNS resolution.

Another common use of this variant is to control routing between the separate address spaces used by each interface. This is often used for PC Server based firewalls.

Multihoming caveats

While multihoming is generally used to eliminate network connectivity as a potential single point of failure (SPOF), certain implementation caveats apply which can affect the success of such a strategy.

In particular, each of the following items must be addressed in order to eliminate the network SPOF:

- **Upstream connectivity**: A given network operations center must have multiple upstream links to *independent* providers. Furthermore, to lessen the possibility of simultaneous damage to all upstream links, the physical location of each of these upstream links should be physically diverse: far enough apart that a piece of machinery (such as a backhoe) won't accidentally sever all connections at the same time.
- **Routers**: Routers and switches must be positioned such that no single piece of network hardware controls all network access to a given host. In particular, it is not uncommon to see multiple Internet uplinks all converge on a

single edge router. In such a configuration, the loss of that single router disconnects the Internet uplink, despite the fact that multiple ISPs are otherwise in use.

- **Host connectivity**: A "reliable" host must be connected to the network over multiple network interfaces, each connected to a separate router or switch. Alternatively, and preferably, the function of a given host could be duplicated across multiple computers, each of which is connected to a different router or switch.
- **Referencing Entities**: Not only must a host be accessible, but in many cases it must also be "referenced" to be useful. For most servers, this means in particular that the name resolution to that server be functional. For example, if the failure of a single element blocks users from properly resolving the DNS name of that server, then the server is effectively inaccessible, despite its otherwise connected state.

The elimination of a single point of failure is achieved only when each component that could potentially fail is duplicated.

IPv4 multihoming

In order to be multihomed, a network must have its own public IP address range and an AS number. Then a connection to two (or more) separate ISPs is established. The routing over these connections is normally controlled by a BGP enabled router.

In the case where one outgoing link from the multihomed network fails, outgoing traffic will automatically be routed via one of the remaining links. More importantly, other networks will be notified, through BGP updates of the multihomed network routes, of the need to route incoming traffic via another ISP and link.

A key pitfall in multihoming is that two apparently independent links, from completely different ISPs may actually share a common transmission line and/or edge router. This will form a single point of failure and considerably reduce the reliability benefits from multihoming.

Another problem to look out for is that multihoming too small a network may not be effective since route filtering is very common among BGP users and smaller prefixes may be filtered out. This will make multihoming fail.

IPv6 multihoming

Multihoming in the next-generation Internet Protocol (IPv6) is not yet standardized, as discussions about the various possible approaches to multihoming are still unresolved.

Current solutions

- Provider Independent Address Space is available in IPv6. This technique has the advantage of working like IPv4, supporting traffic balancing across multiple providers, and maintaining existing TCP and UDP sessions through cut-overs. Critics say that the increased size of routing tables needed to handle multi-homing in this way will overwhelm current router hardware. Proponents say that new hardware will be able to handle the increase due to cheaper memory, which drops in price according to Moore's law. Proponents also say this is the only viable solution right now, and the worse is better philosophy supports the idea that it is better to deploy an imperfect solution now than a perfect solution after it's too late.

Because many ISPs filter out route announcements with small prefixes, this will generally require a large "ISP-sized" IP allocation, such as a /32, to ensure global reachability. Using such large prefixes is an inefficient use of IPv6's address space; there are only about 4 billion /32 prefixes. However, from a pragmatic perspective, allocating a /32 is equivalent in global address space cost to allocating a single IPv4 address, and this may be acceptable if, as seems to be likely for the foreseeable future, the number of multihomed sites can be numbered only in the millions, as opposed to the many billions of non-multihomed endpoints which are anticipated to comprise the vast majority of IPv6 endpoints. Some RIRs such as RIPE has started to allocate /48 from a specific prefix for this purpose. RIPE allocate IPv6 PI /48s or shorter from 2001:0678::/29.

Other current possibilities

- Automated renumbering. If one uplink goes down, all addresses in the network will be renumbered into a new /48 subnet. DNS and firewall records must be updated to redirect traffic to a different /48 subnet. This renumbering will break live TCP and UDP sessions.
- Maintaining multiple simultaneous sets of host addresses, from different upstream /48's for each host, and using multiple AAAA records. This works in most cases, but has the disadvantage that DNS and firewall records must be updated to redirect traffic to the correct set of IP addresses if one of the links goes down. Since this also changes IP addresses on failure, it will still break live TCP and UDP sessions.

Potential future solutions

- Site Multihoming by IPv6 Intermediation

  Con: Operators don't believe in it

  Con: Multihoming configured on host stack, no support for middleboxes (firewalls, proxies)
- Host Identity Protocol

  Con: Not deployed

  Con: All applications need to support it

  Pro: Free certificates
- Stream Control Transmission Protocol

  Con: All applications need to support it
- Locator/Identifier Separation Protocol

External links

IPv4 multihoming:

- O'Reilly article on BGP Multihoming [1]
- Elfiq Networks' introduction to multihoming though link balancer appliances without BGP [2]
- Cisco multihoming configuration example [3]
- Linux Multihoming configuration example [4]
- Windows Multihoming example for Single Link, Multiple IP address [5] (Link down as per 2009 Jan 28, see the internet archive [6] for a cached version)

IPv6 multihoming:

- Ecessa article on multihoming [7]
- Old IETF IPv6 multihoming working group [8]
- Current IETF IPv6 multihoming working group [9]
- Internet-Draft: Analysis of IPv6 Multihoming Scenarios [10]

References

[1] http://www.oreillynet.com/pub/a/network/2002/08/12/multihoming.html
[2] http://www.elfiq.com/how
[3] http://www.cisco.com/en/US/tech/tk365/technologies_configuration_example09186a008009456d.shtml
[4] http://tetro.net/misc/multilink.html
[5] http://www.uit.co.uk/practical-tcpip/w-ipaliasw.pdf
[6] http://web.archive.org/web/20070204050951/http://www.uit.co.uk/practical-tcpip/w-ipaliasw.pdf
[7] http://www.ecessa.com/pages/solutions/solutions_technology_landing.php
[8] http://ops.ietf.org/multi6/
[9] http://www.ietf.org/html.charters/shim6-charter.html
[10] http://www.consulintel.euro6ix.org/ietf/draft-palet-multi6-scenarios-00.txt

Network address translation

In computer networking, **network address translation** (**NAT**) is the process of modifying IP address information in IP packet headers while in transit across a traffic routing device.

The simplest type of NAT provides a one to one translation of IP addresses. RFC 2663 refers to this type of NAT as **basic NAT**. It is often also referred to as **one-to-one NAT**. In this type of NAT only the IP addresses, IP header checksum and any higher level checksums that include the IP address need to be changed. The rest of the packet can be left untouched (at least for basic TCP/UDP functionality, some higher level protocols may need further translation). Basic NATs can be used when there is a requirement to interconnect two IP networks with incompatible addressing.

However it is common to hide an entire IP address space, usually consisting of private IP addresses, behind a single IP address (or in some cases a small group of IP addresses) in another (usually public) address space. To avoid ambiguity in the handling of returned packets, a one-to-many NAT must alter higher level information such as TCP/UDP ports in outgoing communications and must maintain a translation table so that return packets can be correctly translated back. RFC 2663 uses the term **NAPT** (**network address and port translation**) for this type of NAT. Other names include **PAT** (**port address translation**), **IP masquerading**, **NAT Overload** and **many-to-one NAT**. Since this is the most common type of NAT it is often referred to simply as NAT.

As described, the method enables communication through the router only when the conversation originates in the masqueraded network, since this establishes the translation tables. For example, a web browser in the masqueraded network can browse a website outside, but a web browser outside could not browse a web site in the masqueraded network. However, most NAT devices today allow the network administrator to configure translation table entries for permanent use. This feature is often referred to as "static NAT" or port forwarding and allows traffic originating in the "outside" network to reach designated hosts in the masqueraded network.

In the mid-1990s NAT became a popular tool for alleviating the consequences of IPv4 address exhaustion.[1] It has become a common, indispensable feature in routers for home and small-office Internet connections. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address.

Network address translation has serious drawbacks on the quality of Internet connectivity and requires careful attention to the details of its implementation. In particular all types of NAT break the originally envisioned model of IP end-to-end connectivity across the Internet and NAPT makes it difficult for systems behind a NAT to accept incoming communications. As a result, NAT traversal methods have been devised to alleviate the issues encountered.

One to many NATs

The majority of NATs map multiple private hosts to one publicly exposed IP address. In a typical configuration, a local network uses one of the designated "private" IP address subnets (RFC 1918). A router on that network has a private address in that address space. The router is also connected to the Internet with a "public" address assigned by an Internet service provider. As traffic passes from the local network to the Internet, the source address in each packet is translated on the fly from a private address to the public address. The router tracks basic data about each active connection (particularly the destination address and port). When a reply returns to the router, it uses the connection tracking data it stored during the outbound phase to determine the private address on the internal network to which to forward the reply.

All Internet packets have a source IP address and a destination IP address. Typically packets passing from the private network to the public network will have their source address modified while packets passing from the public network back to the private network will have their destination address modified. More complex configurations are also possible.

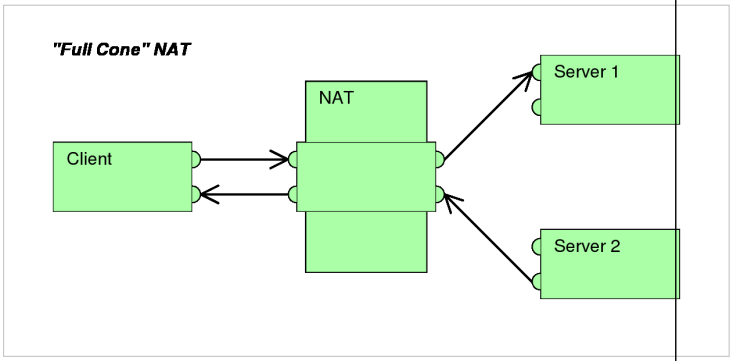
To avoid ambiguity in how to translate returned packets, further modifications to the packets are required. The vast bulk of Internet traffic is TCP and UDP packets and for these protocols the port numbers are changed so that the combination of IP and port information on the returned packet can be unambiguously mapped to the corresponding private address and port information. Protocols not based on TCP or UDP require other translation techniques. ICMP packets typically relate to an existing connection and need to be mapped using the same IP and port mappings as that connection.

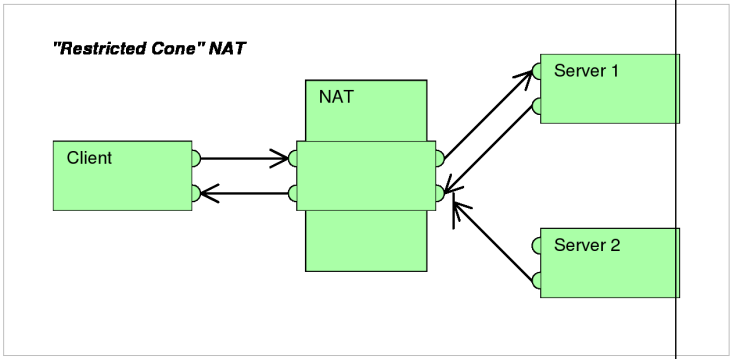
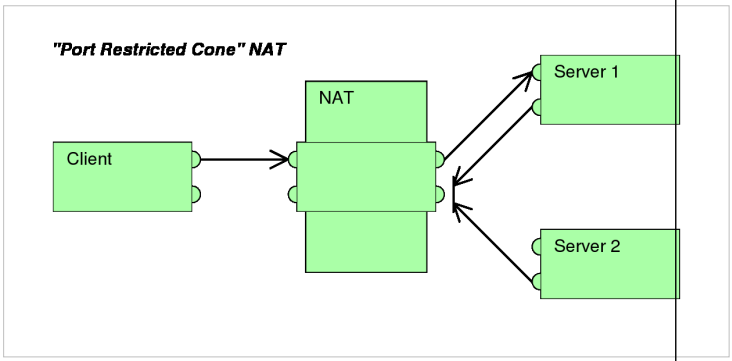
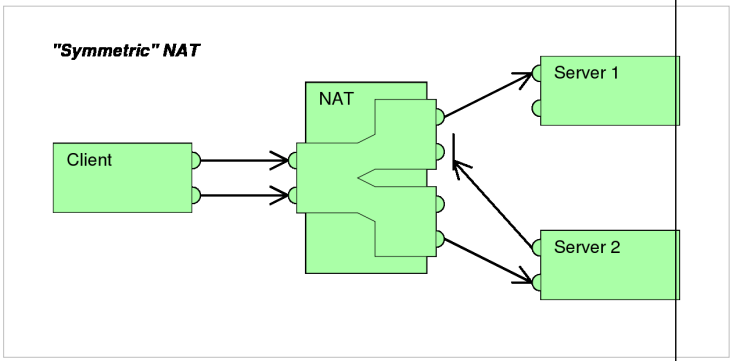
Methods of Port translation

There are several ways of implementing network address and port translation. In some application protocols that use IP address information, the application running on a node in the masqueraded network needs to determine the external address of the NAT, i.e., the address that its communication peers detect, and, furthermore, often needs to examine and categorize the type of mapping in use. Usually this is done because it is desired to set up a direct communications path (either to save the cost of taking the data via a server or to improve performance) between two clients both of which are behind separate NATs. For this purpose, the Simple traversal of UDP over NATs (STUN) protocol was developed (RFC 3489, March 2003). It classified NAT implementation as *full cone NAT*, *(address) restricted cone NAT*, *port restricted cone NAT* or *symmetric NAT* and proposed a methodology for testing a device accordingly. However, these procedures have since been deprecated from standards status, as the methods have proven faulty and inadequate to correctly assess many devices. New methods have been standardized in RFC 5389 (October 2008) and the STUN acronym now represents the new title of the specification: *Session Traversal Utilities for NAT*.

Full-cone NAT, also known as *one-to-one NAT*

- Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort.
- *Any external host* can send packets to iAddr:iPort by sending packets to eAddr:ePort.



(Address) restricted cone NAT • Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort. • An external host (*hAddr:any*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously sent a packet to hAddr:*any*. "Any" means the port number doesn't matter.	
Port-restricted cone NAT Like an address restricted cone NAT, but the restriction includes port numbers. • Once an internal address (iAddr:iPort) is mapped to an external address (eAddr:ePort), any packets from iAddr:iPort will be sent through eAddr:ePort. • An external host (*hAddr:hPort*) can send packets to iAddr:iPort by sending packets to eAddr:ePort only if iAddr:iPort has previously sent a packet to hAddr:hPort.	
Symmetric NAT • Each request from the same internal IP address and port to a specific destination IP address and port is mapped to a unique external source IP address and port, if the same internal host sends a packet even with the same source address and port but to a different destination, a different mapping is used. • Only an external host that receives a packet from an internal host can send a packet back.	

This terminology has been the source of much confusion, as it has proven inadequate at describing real-life NAT behavior.[2] Many NAT implementations combine these types, and it is therefore better to refer to specific individual NAT behaviors instead of using the Cone/Symmetric terminology. Especially, most NAT translators combine *symmetric NAT* for outgoing connections with *static port mapping*, where incoming packets to the external address and port are redirected to a specific internal address and port. Some products can redirect packets to several internal hosts, e.g. to divide the load between a few servers. However, this introduces problems with more sophisticated communications that have many interconnected packets, and thus is rarely used.

Type of NAT and NAT Traversal

The NAT traversal problem arises when two peers behind distinct NAT try to communicate. One way to solve this problem is to use port forwarding, another way is to use various NAT traversal techniques. The most popular technique for TCP NAT traversal is TCP hole punching, which requires the NAT to follow the *port preservation* design for TCP, as explained below.

Many NAT implementations follow the *port preservation* design especially for TCP, which is to say that they use the same values as internal and external port numbers. NAT *port preservation* for outgoing TCP connections is especially important for TCP NAT traversal, because programs usually bind distinct TCP sockets to ephemeral ports for distinct TCP connections, rendering NAT port prediction impossible for TCP.

On the other hand, for UDP, NATs do not need to have *port preservation* because applications usually reuse the same UDP socket to send packets to distinct hosts, making port prediction straightforward, as it is the same source port for each packet.

Furthermore, *port preservation* in NAT for TCP allows P2P protocols to offer less complexity and less latency because there is no need to use a third party to discover the NAT port.[3]

However, if two internal hosts attempt to communicate with the same external host using the same port number, the external port number used by the second host will be chosen at random. Such NAT will be sometimes perceived as *(address) restricted cone NAT* and other times as *symmetric NAT*.

Recent studies have shown that roughly 70% of clients in P2P networks employ some form of NAT.[4]

Implementation

Establishing Two-Way Communication

Every TCP and UDP packet contains both a source IP address and source port number as well as a destination IP address and destination port number. The port address/IP address pair forms a socket. In particular, the source port address and source IP address form the source socket.

For publicly accessible services such as web servers and mail servers the port number is important. For example, port 80 connects to the web server software and port 25 to a mail server's SMTP daemon. The IP address of a public server is also important, similar in global uniqueness to a postal address or telephone number. Both IP address and port must be correctly known by all hosts wishing to successfully communicate.

Private IP addresses as described in RFC 1918 are significant only on private networks where they are used, which is also true for host ports. Ports are unique endpoints of communication on a host, so a connection through the NAT device is maintained by the combined mapping of port and IP address.

PAT resolves conflicts that would arise through two different hosts using the same source port number to establish unique connections at the same time.

An Analogy

A NAT device is similar to a phone system at an office that has one public telephone number and multiple extensions. Outbound phone calls made from the office all appear to come from the same telephone number. However, an incoming call that does not specify an extension cannot be transferred to an individual inside the office. In this scenario, the office is a private LAN, the main phone number is the public IP address, and the individual extensions are unique port numbers.[5]

Translation of the Endpoint

With NAT, all communication sent to external hosts actually contain the *external* IP address and port information of the NAT device instead of internal host IPs or port numbers.

- When a computer on the private (internal) network sends a packet to the external network, the NAT device replaces the internal IP address in the source field of the packet header (*sender's address*) with the external IP address of the NAT device. PAT may then assign the connection a port number from a pool of available ports, inserting this port number in the source port field (much like the *post office box number*), and forwards the packet to the external network. The NAT device then makes an entry in a translation table containing the internal IP address, original source port, and the translated source port. Subsequent packets from the same connection are translated to the same port number.
- The computer receiving a packet that has undergone NAT establishes a connection to the port and IP address specified in the altered packet, oblivious to the fact that the supplied address is being translated (analogous to using a *post office box number*).
- A packet coming from the external network is mapped to a corresponding internal IP address and port number from the translation table, replacing the external IP address and port number in the incoming packet header (similar to the translation from *post office box number* to *street address*). The packet is then forwarded over the inside network. Otherwise, if the destination port number of the incoming packet is not found in the translation table, the packet is dropped or rejected because the PAT device doesn't know where to send it.

NAT will only translate IP addresses and ports of its internal hosts, hiding the true endpoint of an internal host on a private network.

Visibility of Operation

NAT operation is typically transparent to both the internal and external hosts.

Typically the internal host is aware of the true IP address and TCP or UDP port of the external host. Typically the NAT device may function as the default gateway for the internal host. However the external host is only aware of the public IP address for the NAT device and the particular port being used to communicate on behalf of a specific internal host.

NAT and TCP/UDP

"Pure NAT", operating on IP alone, may or may not correctly parse protocols that are totally concerned with IP information, such as ICMP, depending on whether the payload is interpreted by a host on the "inside" or "outside" of translation. As soon as the protocol stack is traversed, even with such basic protocols as TCP and UDP, the protocols will break unless NAT takes action beyond the network layer.

IP packets have a checksum in each packet header, which provides error detection only for the header. IP datagrams may become fragmented and it is necessary for a NAT to reassemble these fragments to allow correct recalculation of higher-level checksums and correct tracking of which packets belong to which connection.

The major transport layer protocols, TCP and UDP, have a checksum that covers all the data they carry, as well as the TCP/UDP header, plus a "pseudo-header" that contains the source and destination IP addresses of the packet carrying the TCP/UDP header. For an originating NAT to pass TCP or UDP successfully, it must recompute the TCP/UDP header checksum based on the translated IP addresses, not the original ones, and put that checksum into the TCP/UDP header of the first packet of the fragmented set of packets. The receiving NAT must recompute the IP checksum on every packet it passes to the destination host, and also recognize and recompute the TCP/UDP header using the retranslated addresses and pseudo-header. This is not a completely solved problem. One solution is for the receiving NAT to reassemble the entire segment and then recompute a checksum calculated across all packets.

The originating host may perform Maximum transmission unit (MTU) path discovery to determine the packet size that can be transmitted without fragmentation, and then set the *don't fragment* (DF) bit in the appropriate packet header field.

Destination network address translation (DNAT)

DNAT is a technique for transparently changing the destination IP address of an en-route packet and performing the inverse function for any replies. Any router situated between two endpoints can perform this transformation of the packet.

DNAT is commonly used to publish a service located in a private network on a publicly accessible IP address. This use of DNAT is also called port forwarding, or DMZ when used on an entire server, which becomes exposed to the WAN, becoming analogous to an undefended military demilitarised zone (DMZ).

SNAT

The meaning of the term *SNAT* varies by vendor. Many vendors have proprietary definitions for *SNAT*. A common expansion is *source NAT*, the counterpart of *destination NAT* (*DNAT*). Microsoft uses the acronym for *Secure NAT*, in regard to the ISA Server. For Cisco Systems, *SNAT* means *stateful NAT*.

Secure network address translation

In computer networking, the process of network address translation done in a secure way involves rewriting the source and/or destination addresses of IP packets as they pass through a router or firewall.

Dynamic network address translation

Dynamic NAT, just like static NAT, is not common in smaller networks but is found within larger corporations with complex networks. The way dynamic NAT differs from static NAT is that where static NAT provides a one-to-one internal to public static IP address mapping, dynamic NAT doesn't make the mapping to the public IP address static and usually uses a group of available public IP addresses.

Applications affected by NAT

Some Application Layer protocols (such as FTP and SIP) send explicit network addresses within their application data. FTP in active mode, for example, uses separate connections for control traffic (commands) and for data traffic (file contents). When requesting a file transfer, the host making the request identifies the corresponding data connection by its network layer and transport layer addresses. If the host making the request lies behind a simple NAT firewall, the translation of the IP address and/or TCP port number makes the information received by the server invalid. The Session Initiation Protocol (SIP) controls many Voice over IP (VoIP) calls, and suffers the same problem. SIP and SDP may use multiple ports to set up a connection and transmit voice stream via RTP. IP addresses and port numbers are encoded in the payload data and must be known prior to the traversal of NATs. Without special techniques, such as STUN, NAT behavior is unpredictable and communications may fail.

Application layer gateway (ALG) software or hardware may correct these problems. An ALG software module running on a NAT firewall device updates any payload data made invalid by address translation. ALGs obviously need to understand the higher-layer protocol that they need to fix, and so each protocol with this problem requires a separate ALG. For example, on many Linux systems, there are kernel modules called *connection trackers* which serve to implement ALGs. However, ALG does not work if the control channel is encrypted (e.g. FTPS).

Another possible solution to this problem is to use NAT traversal techniques using protocols such as STUN or ICE, or proprietary approaches in a session border controller. NAT traversal is possible in both TCP- and UDP-based

applications, but the UDP-based technique is simpler, more widely understood, and more compatible with legacy NATs. In either case, the high level protocol must be designed with NAT traversal in mind, and it does not work reliably across symmetric NATs or other poorly-behaved legacy NATs.

Other possibilities are UPnP (Universal Plug and Play) or NAT-PMP (NAT Port Mapping Protocol), but these require the cooperation of the NAT device.

Most traditional client-server protocols (FTP being the main exception), however, do not send layer 3 contact information and therefore do not require any special treatment by NATs. In fact, avoiding NAT complications is practically a requirement when designing new higher-layer protocols today (e.g. the use of SFTP instead of FTP).

NATs can also cause problems where IPsec encryption is applied and in cases where multiple devices such as SIP phones are located behind a NAT. Phones which encrypt their signaling with IPsec encapsulate the port information within an encrypted packet, meaning that NA(P)T devices cannot access and translate the port. In these cases the NA(P)T devices revert to simple NAT operation. This means that all traffic returning to the NAT will be mapped onto one client causing service to more than one client "behind" the NAT to fail. There are a couple of solutions to this problem: one is to use TLS, which operates at level 4 in the OSI Reference Model and therefore does not mask the port number; another is to encapsulate the IPsec within UDP - the latter being the solution chosen by TISPAN to achieve secure NAT traversal.

The DNS protocol vulnerability announced by Dan Kaminsky on July 8, 2008 is indirectly affected by NAT port mapping. To avoid DNS server cache poisoning, it is highly desirable to not translate UDP source port numbers of outgoing DNS requests from a DNS server which is behind a firewall which implements NAT. The recommended work-around for the DNS vulnerability is to make all caching DNS servers use randomized UDP source ports. If the NAT function de-randomizes the UDP source ports, the DNS server will be made vulnerable.

Advantages of PAT

In addition to the advantages provided by NAT:

- PAT (Port Address Translation) allows many internal hosts to share a single external IP address.
- Users who do not require support for inbound connections do not consume public IP addresses.

Drawbacks

The primary purpose of IP-masquerading NAT is that it has been a practical solution to the impending exhaustion of IPv4 address space. Even large networks can be connected to the Internet with as little as a single IP address. The more common arrangement is having machines that require end-to-end connectivity supplied with a routable IP address, while having machines that do not provide services to outside users behind NAT with only a few IP addresses used to enable Internet access, however, this brings some problems, outlined below.

Some[6] have also called this exact feature a major drawback, since it delays the need for the implementation of IPv6:

> "[...] it is possible that its [NAT's] widespread use will significantly delay the need to deploy IPv6. [...] It is probably safe to say that networks would be better off without NAT [...]"

Hosts behind NAT-enabled routers do not have end-to-end connectivity and cannot participate in some Internet protocols. Services that require the initiation of TCP connections from the outside network, or stateless protocols such as those using UDP, can be disrupted. Unless the NAT router makes a specific effort to support such protocols, incoming packets cannot reach their destination. Some protocols can accommodate one instance of NAT between participating hosts ("passive mode" FTP, for example), sometimes with the assistance of an application-level gateway (see below), but fail when both systems are separated from the Internet by NAT. Use of NAT also complicates tunneling protocols such as IPsec because NAT modifies values in the headers which interfere with the integrity checks done by IPsec and other tunneling protocols.

End-to-end connectivity has been a core principle of the Internet, supported for example by the Internet Architecture Board. Current Internet architectural documents observe that NAT is a violation of the End-to-End Principle, but that NAT does have a valid role in careful design.[7] There is considerably more concern with the use of IPv6 NAT, and many IPv6 architects believe IPv6 was intended to remove the need for NAT.[8]

Because of the short-lived nature of the stateful translation tables in NAT routers, devices on the internal network lose IP connectivity typically within a very short period of time unless they implement NAT keep-alive mechanisms by frequently accessing outside hosts. This dramatically shortens the power reserves on battery-operated hand-held devices and has thwarted more widespread deployment of such IP-native Internet-enabled devices.

Some Internet service providers (ISPs), especially in India, Russia, parts of Asia and other "developing" regions provide their customers only with "local" IP addresses, due to a limited number of external IP addresses allocated to those entities. Thus, these customers must access services external to the ISP's network through NAT. As a result, the customers cannot achieve true end-to-end connectivity, in violation of the core principles of the Internet as laid out by the Internet Architecture Board.

- Scalability - An implementation that only tracks ports can be quickly depleted by internal applications that use multiple simultaneous connections (such as an HTTP request for a web page with many embedded objects). This problem can be mitigated by tracking the destination IP address in addition to the port (thus sharing a single local port with many remote hosts), at the expense of implementation complexity and CPU/memory resources of the translation device.
- Firewall complexity - Because the internal addresses are all disguised behind one publicly-accessible address, it is impossible for external hosts to initiate a connection to a particular internal host without special configuration on the firewall to forward connections to a particular port. Applications such as VOIP, videoconferencing, and other peer-to-peer applications must use NAT traversal techniques to function.

Specifications

IEEE [9] Reverse Address and Port Translation (RAPT, or RAT) allows a host whose real IP address is changing from time to time to remain reachable as a server via a fixed home IP address. In principle, this should allow setting up servers on DHCP-run networks. While not a perfect mobility solution, RAPT together with upcoming protocols like DHCP-DDNS, it may end up becoming another useful tool in the network admin's arsenal.

IETF [10] *RAPT* (IP Reachability Using Twice Network Address and Port Translation) The RAT device maps an IP datagram to its associated CN and 0MN by using three additional fields: the IP protocol type number and the transport layer source and destination connection identifiers (e.g. TCP port number or ICMP echo request/reply ID field).

Cisco *RAPT* implementation is PAT (Port Address Translation) or overloading , and maps multiple private IP addresses to a single public IP address. Multiple addresses can be mapped to a single address because each private address is tracked by a port number. PAT uses unique source port numbers on the inside global IP address to distinguish between translations. The port number is encoded in 16 bits. The total number of internal addresses that can be translated to one external address could theoretically be as high as 65,536 per IP address. Realistically, the number of ports that can be assigned a single IP address is around 4000. PAT will attempt to preserve the original source port. If this source port is already used, PAT will assign the first available port number starting from the beginning of the appropriate port group 0-511, 512-1023, or 1024-65535. When there are no more ports available and there is more than one external IP address configured, PAT moves to the next IP address to try to allocate the original source port again. This process continues until it runs out of available ports and external IP addresses.

3COM **U.S. Patent 6055236** [11] (Method and system for locating network services with distributed network address translation) Methods and system for locating network services with distributed network address translation. Digital certificates are created that allow an external network device on an external network, such as the Internet, to request a service from an internal network device on an internal distributed network address translation network, such as a

stub local area network. The digital certificates include information obtained with a Port Allocation Protocol used for distributed network address translation. The digital certificates are published on the internal network so they are accessible to external network devices. An external network device retrieves a digital certificate, extracts appropriate information, and sends a service request packet to an internal network device on an internal distributed network address translation network. The external network device is able to locate and request a service from an internal network device. An external network device can also request a security service, such as an Internet Protocol security ("IPsec") service from an internal network device. The external network device and the internal network device can establish a security service (e.g., Internet Key Exchange protocol service). The internal network device and external network device can then establish a Security Association using Security Parameter Indexes ("SPI") obtained using a distributed network address translation protocol. External network devices can request services, and security services on internal network devices on an internal distribute network address translation network that were previously unknown and unavailable to the external network devices.

Examples of NAT software

- Internet Connection Sharing (ICS): Windows NAT+DHCP since W98SE
- WinGate: like ICS plus lots of control
- iptables: the Linux packet filter and NAT (interface for NetFilter)
- IPFilter: Solaris, NetBSD, FreeBSD, xMach.
- PF (firewall): The OpenBSD Packet Filter.
- Netfilter Linux packet filter framework

References

[1] www.tcpipguide.com/free/t_IPNetworkAddressTranslationNATProtocol.htm (http://www.tcpipguide.com/free/t_IPNetworkAddressTranslationNATProtocol.htm)

[2] François Audet; and Cullen Jennings (January 2007) (text). *RFC 4787 Network Address Translation (NAT) Behavioral Requirements for Unicast UDP* (http://www.ietf.org/rfc/rfc4787.txt). IETF. . Retrieved 2007-08-29.

[3] "Characterization and Measurement of TCP Traversal through NATs and Firewalls" (http://nutss.gforge.cis.cornell.edu/pub/imc05-tcpnat/). December 2006. .

[4] "Illuminating the shadows: Opportunistic network and web measurement" (http://illuminati.coralcdn.org/stats/). December 2006. .

[5] "The Audio over IP Instant Expert Guide" (http://www.tieline.com/Downloads/Audio-over-IP-Instant-Expert-Guide-v1.pdf). Tieline. January 2010. . Retrieved 2011-08-19.

[6] Larry L. Peterson; and Bruce S. Davie; *Computer Networks: A Systems Approach*, Morgan Kaufmann, 2003, pp. 328-330, ISBN 1-55860-832-X

[7] R. Bush; and D. Meyer; RFC 3439, *Some Internet Architectural Guidelines and Philosophy* (http://www.ietf.org/rfc/rfc3439.txt), December 2002

[8] G. Van de Velde *et al.*; RFC 4864, *Local Network Protection for IPv6* (http://tools.ietf.org/rfc/rfc4864.txt), May 2007

[9] http://ieeexplore.ieee.org/iel4/6056/16183/00749275.pdf

[10] http://www3.ietf.org/proceedings/99nov/I-D/draft-ietf-nat-rnat-00.txt

[11] http://www.google.com/patents?vid=6055236

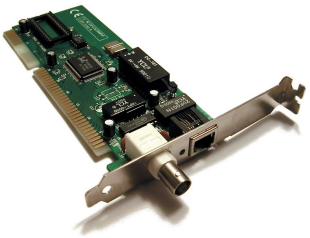
External links

- NAT-Traversal Test and results (http://nattest.net.in.tum.de)
- Characterization of different TCP NATs (http://nutss.net/pub/imc05-tcpnat/) – Paper discussing the different types of NAT
- Anatomy: A Look Inside Network Address Translators – Volume 7, Issue 3, September 2004 (http://www.cisco.com/en/US/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html)
- Jeff Tyson, HowStuffWorks: *How Network Address Translation Works* (http://computer.howstuffworks.com/nat.htm/printable)

- NAT traversal techniques in multimedia Networks (http://www.newport-networks.com/whitepapers/nat-traversal1.html) – White Paper from Newport Networks
- Peer-to-Peer Communication Across Network Address Translators (http://www.brynosaurus.com/pub/net/p2pnat/) (PDF) (http://www.brynosaurus.com/pub/net/p2pnat.pdf) – NAT traversal techniques for UDP and TCP
- http://www.zdnetasia.com/insight/network/0,39044847,39050002,00.htm
- RFCs
  - RFC 1631 (Status: Obsolete) - The IP Network Address Translator (NAT)
  - RFC 1918 - Address Allocation for Private Internets
  - RFC 3022 (Status: Informational) – Traditional IP Network Address Translator (Traditional NAT)
  - RFC 4008 (Status: Standards Track) – Definitions of Managed Objects for Network Address Translators (NAT)
  - RFC 5128 (Status: Informational) - State of Peer-to-Peer (P2P) Communications across Network Address Translators (NATs)
  - RFC 4966 (Status: Informational) - Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status
- *Speak Freely* End of Life Announcement (http://www.fourmilab.ch/speakfree/unix/) – John Walker's discussion of why he stopped developing a famous program for free Internet communication, part of which is directly related to NAT
- natd (http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-natd.html)
- SNAT, DNAT and OCS2007R2 (http://www.cainetworks.com/support/training/snat-dnat-ocs.html) – discussing the SNAT in Microsoft OCS 2007R2
- Alternative Taxonomy (Part of the documentation for the IBM iSeries)
  - Static NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwstatic.htm)
  - Dynamic NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwdynamic.htm)
  - Masquerade NAT (http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzajw/rzajwaddmasq.htm)
- Network Address Translation - NAT (http://blog.ipexpert.com/2009/09/07/network-address-translation-nat/)
- Cisco Systems
  - Document ID 6450: How NAT Works (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml)
  - Document ID 26704: Network Address Translation (NAT) FAQ (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_q_and_a_item09186a00800e523b.shtml)
  - White Paper: Cisco IOS Network Address Translation Overview (http://www.cisco.com/en/US/technologies/tk648/tk361/tk438/technologies_white_paper09186a0080091cb9.html)
  - Cisco IOS NAT Commands Cisco IOS commands (http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/cs/csprtd/csprtd11/csnat.htm)
  - Animation Cisco NAT sample (http://www.cisco.com/image/gif/paws/6450/nat.swf)

Network interface controller

Network Interface Card (NIC)



A 1990s Ethernet network interface controller card which connects to the motherboard via the now-obsolete ISA bus. This combination card features both a (now obsolete) bayonet cap BNC connector (left) for use in coaxial-based 10base2 networks and an 8P8C connector (right) for use in twisted pair-based 10baseT networks. (The ports could not be used simultaneously.)

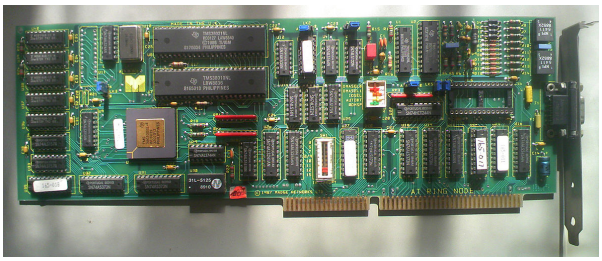
Connects to	Motherboard via one of: • Integrated • PCI Connector • ISA Connector • PCI-E • FireWire • USB Network via one of: • Fast Ethernet • Gigabit Ethernet • Optical fiber • Token ring
Speeds	10 Mbit/s 100 Mbit/s 1000 Mbit/s up to 160 Gbit/s
Common manufacturers	Novell Intel Realtek Others

A **network interface controller** (also known as a **network interface card**, **network adapter**, **LAN adapter** and by similar terms) is a computer hardware component that connects a computer to a computer network.

Whereas network interface controllers were commonly implemented on expansion cards that plug into a computer bus, the low cost and ubiquity of the Ethernet standard means that most newer computers have a network interface built into the motherboard.

Purpose

The network controller implements the electronic circuitry required to communicate using a specific physical layer and data link layer standard such as Ethernet, Wi-Fi, or Token Ring. This provides a base for a full network protocol stack, allowing communication among small groups of computers on the same LAN and large-scale network communications through routable protocols, such as IP.



Madge 4/16 Mbit/s TokenRing ISA-16 NIC

Although other network technologies exist (e.g. token ring), Ethernet has achieved near-ubiquity since the mid-1990s.

Every Ethernet network controller has a unique 48-bit serial number called a MAC address, which is stored in read-only memory. Every computer on an Ethernet network must have at least one controller. Normally it is safe to assume that no two network controllers will share the same address, because controller vendors purchase blocks of addresses from the Institute of Electrical and Electronics Engineers (IEEE) and assign a unique address to each controller at the time of manufacture.[1]

The NIC allows computers to communicate over a computer network. It is both an OSI layer 1 (physical layer) and layer 2 (data link layer) device, as it provides physical access to a networking medium and provides a low-level addressing system through the use of MAC addresses. It allows users to connect to each other either by using cables or wirelessly.

Implementation

Whereas network controllers used to operate on expansion cards that plugged into a computer bus, the low cost and ubiquity of the Ethernet standard means that most newer computers have a network interface built into the motherboard. Newer server motherboards may even have dual network interfaces built-in. The Ethernet capabilities are either integrated into the motherboard chipset or implemented via a low-cost dedicated Ethernet chip, connected through the PCI (or the newer PCI express) bus. A separate network card is not required unless additional interfaces are needed or some other type of network is used.

The NIC may use one or more of four techniques to transfer data:

- Polling is where the CPU examines the status of the peripheral under program control.
- Programmed I/O is where the microprocessor alerts the designated peripheral by applying its address to the system's address bus.
- Interrupt-driven I/O is where the peripheral alerts the microprocessor that it is ready to transfer data.
- Direct memory access is where an intelligent peripheral assumes control of the system bus to access memory directly. This removes load from the CPU but requires a separate processor on the card.

An Ethernet network controller typically has an 8P8C socket where the network cable is connected. Older NICs also supplied BNC, or AUI connections. A few LEDs inform the user of whether the network is active, and whether or not data transmission occurs. Ethernet network controllers typically support 10 Mbit/s Ethernet, 100 Mbit/s Ethernet, and 1000 Mbit/s Ethernet varieties. Such controllers are designated *10/100/1000* - this means they can support a notional maximum transfer rate of 10, 100 or 1000 Megabits per second.

Some products feature NIC partitioning (NPAR).[2]

References

[1] IEEE Registration Authority - IEEE OUI and Company_id Assignments (http://standards.ieee.org/regauth/oui/index.shtml)
[2] For example, "QLogic QLE3240-LR-CK - network adapter" (http://reviews.cnet.com/adapters-nics/qlogic-qle3240-lr-ck/4507-3380_7-34476794.html?tag=mncolBtm;rnav). *cnet Reviews*. CBS Interactive. 2011. . Retrieved 2011-06-23.

Network layer

The **network layer** is **layer 3** of the seven-layer OSI model of computer networking.

The network layer is responsible for packet forwarding including routing through intermediate routers, whereas the data link layer is responsible for media access control, flow control and error checking.

The network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination host via one or more networks while maintaining the quality of service functions.

Functions of the network layer include:

- Connection model: connectionless communication

  For example, IP is connectionless, in that a frame can travel from a sender to a recipient without the recipient having to send an acknowledgement. Connection-oriented protocols exist at other, higher layers of that model.

- Host addressing

  Every host in the network needs to have a unique address which determines where it is. This address will normally be assigned from a hierarchical system, so you can be "Fred Murphy" to people in your house, "Fred Murphy, 1 Main Street" to Dubliners, or "Fred Murphy, 1 Main Street, Dublin" to people in Ireland, or "Fred Murphy, 1 Main Street, Dublin, Ireland" to people anywhere in the world. On the Internet, addresses are known as Internet Protocol (IP) addresses.

- Message forwarding

  Since many networks are partitioned into subnetworks and connect to other networks for wide-area communications, networks use specialized hosts, called gateways or routers to forward packets between networks. This is also of interest to mobile applications, where a user may move from one location to another, and it must be arranged that his messages follow him. Version 4 of the Internet Protocol (IPv4) was not designed with this feature in mind, although mobility extensions exist. IPv6 has a better designed solution.

Within the service layering semantics of the OSI network architecture the network layer responds to service requests from the transport layer and issues service requests to the data link layer.

Protocols

- IPv4/IPv6, Internet Protocol
- DVMRP, Distance Vector Multicast Routing Protocol
- ICMP, Internet Control Message Protocol
- IGMP, Internet Group Multicast Protocol
- PIM-SM, Protocol Independent Multicast Sparse Mode
- PIM-DM, Protocol Independent Multicast Dense Mode
- IPsec, Internet Protocol Security
- IPX, Internetwork Packet Exchange
- RIP, Routing Information Protocol
- DDP, Datagram Delivery Protocol
- BGP, Border Gateway Protocol

Relation to TCP/IP model

The TCP/IP model describes the protocol suite of the Internet (RFC 1122). This model has a layer called the Internet layer, located above the link layer. In many textbooks and other secondary references the Internet layer is equated with OSI's network layer. However, this comparison is misleading as the allowed characteristics of protocols (e.g., whether they are connection-oriented or connection-less) placed into these layers are different in the two models. The Internet layer of TCP/IP is in fact only a subset of functionality of the network layer. It only describes one type of network architecture, the Internet.

In general, direct or strict comparisons between these models should be avoided, since the layering in TCP/IP is not a principal design criterion and the Internet Engineering Task Force (IETF) considers it to be "harmful" (RFC 3439).

References

- RFC 1122
- RFC 3439
- Tanenbaum, Andrew S. (2003). *Computer networks*. Upper Saddle River, New Jersey: Prentice Hall. ISBN 0-13-066102-3.

External links

- OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection [1]PDF (776 KB), Hubert Zimmermann, IEEE Transactions on Communications, vol. 28, no. 4, April 1980, pp. 425-432.

References

[1] http://www.comsoc.org/livepubs/50_journals/pdf/RightsManagement_eid=136833.pdf

Out-of-order delivery

In computer networking, **out-of-order delivery** is the delivery of data packets in a different order from which they were sent. Out-of-order delivery can be caused by packets following multiple paths through a network, or via parallel processing paths within network equipment that are not designed to ensure that packet ordering is preserved. One of the functions of TCP is to prevent the out-of-order delivery of data, either by reassembling packets into order or forcing retries of out-of-order packets.

Packet reordering is a common behavior in real-world networks.

External links

- RFC 4737, *Packet Reordering Metrics*, A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, J. Perser, November 2006
- RFC 5236, *Improved Packet Reordering Metrics*, A. Jayasumana, N. Piratla, T. Banka, A. Bare, R. Whitner, June 2008
- http://kb.pert.geant.net/PERTKB/PacketReordering
- http://www-iepm.slac.stanford.edu/monitoring/reorder/

Packet loss

Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise.

Causes

Packet loss can be caused by a number of factors including signal degradation over the network medium due to multi-path fading, packet drop because of channel congestion [1] [2] , corrupted packets rejected in-transit, faulty networking hardware, faulty network drivers or normal routing routines (such as DSR in ad-hoc networks [3]).

In addition to this, packet loss probability is also affected by signal-to-noise ratio and distance between the transmitter and receiver.

Effects

When caused by network problems, lost or dropped packets can result in highly noticeable performance issues or jitter with streaming technologies, voice over IP, online gaming and videoconferencing, and will affect all other network applications to a degree. [4] However, it is important to note that packet loss does not always indicate a problem. If the latency and the packet loss at the destination hop are acceptable then the hops prior to that one don't matter. [5]

Packet recovery

Some network transport protocols such as TCP provide for reliable delivery of packets. In the event of packet loss, the receiver asks for retransmission or the sender automatically resends any segments that have not been acknowledged. [6] Although TCP can recover from packet loss, retransmitting missing packets causes the throughput of the connection to decrease. This drop in throughput is due to the sliding window protocols used for acknowledgment of received packets. In certain variants of TCP, if a transmitted packet is lost, it will be re-sent

along with every packet that had been sent after it. This retransmission causes the overall throughput of the connection to drop.

Protocols such as UDP provide no recovery for lost packets. Applications that use UDP are expected to define their own mechanisms for handling packet loss.

Acceptable packet loss

"The fraction of lost packets increases as the traffic intensity increases. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss…a lost packet may be retransmitted on an end-to-end basis in order to ensure that all data are[sic] eventually transferred from source to destination." [7] The amount of packet loss that is acceptable depends on the type of data being sent. For example, for Voice over IP traffic, the only effect seen due to the occasional dropped packet is jitter, and therefore "[m]issing one or two packets every now and then will not affect the quality of the conversation. Losses between 5% and 10% of the total packet stream will affect the quality significantly."[8] On the other hand, when transmitting a text document or web page, a single dropped packet could result in losing part of the file, which is where the aforementioned packet retransmission schemes are used. When given a situation where the amount of content due to be pushed through a connection is growing at a rate greater than it is possible to push through that connection, also known as a bottleneck, then there is no other solution than to drop packets.[9] The TCP protocol is designed with a slow-start connection strategy so that excessive packet loss will cause the sender to throttle back and stop flooding the bottleneck point with data (using perceived packet loss as feedback to discover congestion). [10] The data packets will be transmitted over a longer duration.

There are many methods used for determining which packets to drop. Most basic networking equipment will use FIFO queuing for packets waiting to go through the bottleneck and they will drop the packet if the queue is full at the time the packet is received. This type of packet dropping is called tail drop. However, dropping packets when the queue is full is a poor solution for any connection that requires real-time throughput. For these types of connections, quality of service and other methods are applied. In some connections, packets may be intentionally dropped in order to slow down specific services for no other reason than to dissuade users from using those services. For this reason, packet loss is not necessarily an indication of poor connection reliability or a bottleneck.

Packet loss is closely associated with quality of service considerations, and is related to the erlang unit of measure.

References

[1] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 30.
[2] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 30.
[3] Perkins, C. E. (2001). Ad-Hoc Networking. Boston: Addison-Wesley. P 147.
[4] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 602.
[5] "Packet loss or latency at intermediate hops." (http://www.nessoft.com/kb/24) (HTTP). . Retrieved 2007-02-25.
[6] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 242.
[7] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 42-43.
[8] Mansfield, K. C. & Antonakos, J. L. (2010). Computer Networking from LANs to WANs: Hardware, Software, and Security. Boston: Course Technology, Cengage Learning. P501.
[9] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 30.
[10] Kurose, J. F. & Ross, K. W. (2010). Computer Networking: A Top-Down Approach. New York: Addison-Wesley. P 282-283

External links

- Interactive animation of TCP fast retransmit simulation: packet loss detection and recovery. (http://www.visualland.net/tcp_histrory.php?simu=tcp_fast_retransmit&protocol=TCP&title=4.Fast transmit&ctype=1)

Private network

In the Internet addressing architecture, a **private network** is a network that uses private IP address space, following the standards set by RFC 1918 and RFC 4193. These addresses are commonly used for home, office, and enterprise local area networks (LANs), when globally routable addresses are not mandatory, or are not available for the intended network applications. Private IP address spaces were originally defined in an effort to delay IPv4 address exhaustion, but they are also a feature of the next generation Internet Protocol, IPv6.

These addresses are characterized as private because they are not globally delegated, meaning they are not allocated to any specific organization, and IP packets addressed by them cannot be transmitted onto the public Internet. Anyone may use these addresses without approval from a regional Internet registry (RIR). If such a private network needs to connect to the Internet, it must use either a network address translator (NAT) gateway, or a proxy server.

Private IPv4 address spaces

The Internet Engineering Task Force (IETF) has directed the Internet Assigned Numbers Authority (IANA) to reserve the following IPv4 address ranges for private networks, as published in RFC 1918[1] :

RFC1918 name	IP address range	number of addresses	*classful* description	largest CIDR block (subnet mask)	host id size
24-bit block	10.0.0.0 – 10.255.255.255	16,777,216	single class A	10.0.0.0/8 (255.0.0.0)	24 bits
20-bit block	172.16.0.0 – 172.31.255.255	1,048,576	16 contiguous class Bs	172.16.0.0/12 (255.240.0.0)	20 bits
16-bit block	192.168.0.0 – 192.168.255.255	65,536	256 contiguous class Cs	192.168.0.0/16 (255.255.0.0)	16 bits

Classful addressing is obsolete and has not been used in the Internet since the implementation of Classless Inter-Domain Routing (CIDR) starting in 1993. For example, while 10.0.0.0/8 was a single class A network, it is common for organizations to divide it into smaller /16 or /24 networks.

Private IPv6 addresses

The concept of private networks and special address reservation for such networks has been carried over to the next generation of the Internet Protocol, IPv6.

The address block `fc00::/7` has been reserved by IANA as described in RFC 4193. These addresses are called Unique Local Addresses (ULA). They are defined as being unicast in character and contain a 40-bit random number in the routing prefix to prevent collisions when two private networks are interconnected. Despite being inherently *local* in usage, the IPv6 address scope of unique local addresses is global.

A former standard proposed the use of so-called "site-local" addresses in the `fec0::/10` range, but due to major concerns about scalability and the poor definition of what constitutes a *site*, its use has been deprecated since September 2004 by RFC 3879.

Link-local addresses

Another type of private networking uses the *link-local address range*. The validity of link-local addresses is limited to a single *link*; e.g. to all computers connected to a switch, or to one wireless network. Hosts on different sides of a bridge are also on the same link, whereas hosts on different sides of a router are on different links.

IPv4

In IPv4, link-local addresses are codified in RFC 5735 and RFC 3927. Their utility is in self-autoconfiguration by network devices when Dynamic Host Configuration Protocol (DHCP) services are not available and manual configuration by a network administrator is not desirable.

The block 169.254/16 is reserved for this purpose, with the exception of the first and the last /24 subnets in the range. If a host on an IEEE 802 (ethernet) network cannot obtain a network address via DHCP, an address from 169.254.1.0 to 169.254.254.255 may be assigned pseudorandomly. The standard prescribes that address collisions must be handled gracefully.

IPv6

In IPv6, link-local addresses are codified in RFC 4862. Their use is mandatory, and an integral part of the IPv6 standard.

The IPv6 addressing architecture (RFC 4291) sets aside the block `fe80::/10` for IP address autoconfiguration.

Common uses

The most common use of private addresses is in residential networks, since most Internet service providers (ISPs) only allocate a single routable IP address to each residential customer, but many homes have more than one computer or other Internet connected device, such as televisions. In this situation, a network address translator (NAT) gateway is usually used to provide Internet connectivity to multiple hosts.

Private addresses are also commonly used in corporate networks, which for security reasons, are not connected directly to the Internet. Often a proxy, SOCKS gateway, or similar devices are used to provide restricted Internet access to network-internal users.

In both cases, private addresses are often seen as enhancing network security for the internal network, since it is difficult for an Internet host to connect directly to an internal system.

Misrouting

It is common for packets originating in private address spaces to be misrouted onto the Internet. Private networks often do not properly configure DNS services for addresses used internally and attempt reverse DNS lookups for these addresses, causing extra traffic to the Internet root nameservers. The AS112 project attempted to mitigate this load by providing special *blackhole* anycast nameservers for private address ranges which only return negative result codes (*not found*) for these queries.

Organizational edge routers are usually configured to drop ingress IP traffic for these networks, which can occur either by misconfiguration, or from malicious traffic using a spoofed source address. Less commonly, ISP edge routers drop such egress traffic from customers, which reduces the impact to the Internet of such misconfigured or malicious hosts on the customer's network.

Merging private networks

Since the private IPv4 address space is relatively small, many private IPv4 networks use the same address space. This creates a common problem when merging such networks, namely the duplication of addresses on multiple devices. In this case, networks or hosts must be renumbered, often a time-consuming task, or a network address translator must be placed between the networks to masquerade the duplicated addresses.

To mitigate this problem for IPv6, RFC 4193 specifies a large (40-bit) unique Global ID to be pseudo-randomly generated by each organization using Unique Global Addresses. It is very unlikely that two network addresses generated in this way will be the same.

Private use of other reserved addresses

Historically other address blocks than the private address ranges have been reserved for other potential future uses. Some organizations have used them for private networking applications despite official warnings of possible future address collisions. Typically these addresses are not referred to as "reserved." IPv4 addresses 240.0.0.0 to 254.255.255.254 (all addresses in 240.0.0.0/4 except 255.0.0.0/8) are designated for future use and research and development.[2]

RFC References

- RFC 1918 – *"Address Allocation for Private Internets"*
- RFC 2036 – *"Observations on the use of Components of the Class A Address Space within the Internet"*
- RFC 2050 – *"Internet Registry IP Allocation Guidelines"*
- RFC 2101 – *"IPv4 Address Behaviour Today"*
- RFC 2663 – *"IP Network Address Translator (NAT) Terminology and Considerations"*
- RFC 3022 – *"Traditional IP Network Address Translator (Traditional NAT)"*
- RFC 3330 – *"Special-Use IPv4 Addresses"* (superseded)
- RFC 5735 – *"Special-Use IPv4 Addresses"*
- RFC 3879 – *"Deprecating Site Local Addresses"*
- RFC 3927 – *"Dynamic Configuration of IPv4 Link-Local Addresses"*
- RFC 4193 – *"Unique Local IPv6 Unicast Addresses"*

References

[1] "RFC 1918: Address Allocation for Private Internets" (http://tools.ietf.org/html/rfc1918#page-4). February 1996. p. 4. .
[2] "IP" (http://www.computerhope.com/jargon/i/ip.htm). ComputerHope. . Retrieved 2011-03-02.

Route poisoning

Route poisoning is a method to prevent a router from sending packets through a route that has become invalid within computer networks. Distance-vector routing protocols in computer networks use route poisoning to indicate to other routers that a route is no longer reachable and should not be considered from their routing tables. Unlike the *split horizon with poison reverse*, route poisoning provide for sending updates with unreachable hop counts immediately to all the nodes in the network.

When the protocol detects an invalid route, all of the routers in the network are informed that the bad route has an infinite (∞) route metric. This makes all nodes on the invalid route seem infinitely distant, resulting in preventing any of the routers from sending packets over the invalid route. [1]

Some distance-vector routing protocols, such as RIP, use a maximum hop count to determine how many routers traffic must go through to reach the destination. Each route has a hop count number assigned to it which is incremented as the routing information is passed from router to router. A route is considered unreachable if the hop count exceeds the maximum allowed. Route poisoning is a method of quickly forgetting outdated routing information from other router's routing tables by changing its hop count to be unreachable (higher than the maximum number of hops allowed) and sending a routing update. In the case of RIP, the maximum hop count is 15, so to perform route poisoning on a route its hop count is changed to 16, deeming it unreachable, and a routing update is sent.

If these updates are lost, some nodes in the network would not be informed that a route is invalid, so they could attempt to send packets over the bad route and cause a problem known as a routing loop. Therefore route poisoning is used in conjunction with holddowns to keep update messages from falsely reinstating the validity of a bad route. This prevents routing loops, improving the overall efficiency of the network.

References

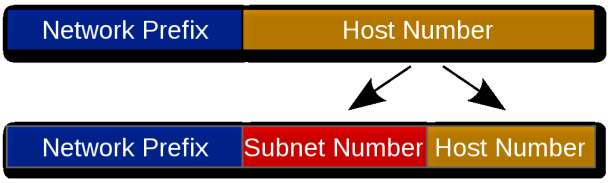
- The TCP-IP Guide, RIP Special Features For Resolving RIP Algorithm Problems, by Charles M. Kozierok [2]
- RFC 1058: Routing Information Protocol, by C. Hedrick, Rutgers University (June 1988) [3]

[1] Wick, Karl (18 April 2007). "What is route poisoning?" (http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci939324,00.html) (in English). . Retrieved 2009-01-23.

[2] http://www.tcpipguide.com/free/t_RIPSpecialFeaturesForResolvingRIPAlgorithmProblems-2.htm

[3] http://www.ietf.org/rfc/rfc1058.txt

Subnetwork



Creating a subnet by dividing the host identifier

A **subnetwork**, or **subnet**, is a logically visible subdivision of an IP network.[1] The practice of dividing a network into subnetworks is called **subnetting**.

All computers that belong to a subnet are addressed with a common, identical, most-significant bit-group in their IP address. This results in the logical division of an IP address into two fields, a network or routing prefix and the rest field. The rest field is a specific identifier for the computer or the network interface.

The routing prefix is expressed in CIDR notation. It is written as the first address of a network followed by the bit-length of the prefix, separated by a slash (/) character. For example, `192.168.1.0/24` is the prefix of the Internet Protocol Version 4 network starting at the given address, having 24 bits allocated for the network prefix, and the remaining 8 bits reserved for host addressing. The IPv6 address specification `2001:db8::/32` is a large network for 2^{96} hosts, having a 32-bit routing prefix. In IPv4 the routing prefix is also specified in the form of the **subnet mask**, which is expressed in quad-dotted decimal representation like an address. For example, `255.255.255.0` is the network mask for the `192.168.1.0/24` prefix.

Traffic between subnetworks is interchanged with special gateway devices called routers; they constitute logical or physical borders between the subnets.

The benefits of subnetting vary with each deployment scenario. In the address allocation architecture of the Internet using Classless Inter-Domain Routing (CIDR) and in large organizations, it is necessary to allocate address space efficiently. It may also enhance routing efficiency, or have advantages in network management when subnetworks are administratively controlled by different entities in a larger organization. Subnets may be arranged logically in a hierarchical architecture, partitioning an organization's network address space into a tree-like routing structure.

Network addressing and routing

Computers participating in a network such as the Internet each have at least one logical address. Usually this address is unique to each device and can either be configured dynamically from a network server, statically by an administrator, or automatically by stateless address autoconfiguration.

An address fulfills the functions of identifying the host and locating it on the network. The most common network addressing architecture is Internet Protocol version 4 (IPv4), but its successor, IPv6, is in early deployment stages. An IPv4 address consists of 32 bits, for human readability written in a form consisting of four decimal octets separated by full stops (dots), called dot-decimal notation. An IPv6 address consists of 128 bits written in a hexadecimal notation and grouping 16 bits separated by colons.

For the purpose of network management, an IP address is logically divided into two fields, a network field and the host field. All hosts on a subnetwork have the same network prefix. This routing prefix occupies the most-significant bits of the address. The number of bits allocated within a network to the internal routing prefix may vary between subnets, depending on the network architecture. While in IPv6 the prefix must consist of a set of contiguous 1-bits, in IPv4 this is not enforced, albeit no efficiency is gained. The rest "host" field is a unique local identification and is either a host number on the local network or an interface identifier.

This logical addressing structure permits the selective routing of IP packets across multiple networks via special gateway computers, called routers, to a destination host if the network prefixes of origination and destination hosts differ, or sent directly to a target host on the local network if they are the same. Routers constitute logical or physical

borders between the subnets, and manage traffic between them. Each subnet is served by a designated default router, but may consist internally of multiple physical Ethernet segments interconnected by network switches or network bridges.

The routing prefix of an address is written in a form identical to that of the address itself. This is called the network mask, or *netmask*, of the address. For example, a specification of the most-significant 18 bits of an IPv4 address, `11111111.11111111.11000000.00000000`, is written as `255.255.192.0`. If this mask designates a subnet within a larger network, it is also called the *subnet mask*. This form of denoting the network mask, however, is only used for IPv4 networks.

The modern standard form of specification of the network prefix, used for both IPv4 and IPv6, counts the number of bits in the prefix and appends that number to the address with a *slash* (/) separator:

- `192.168.0.0`, netmask `255.255.255.0` is written as `192.168.0.0/24`
- In IPv6, `2001:db8::/32` designates the address `2001:db8::` and its network prefix consisting of the most significant 32 bits.

This notation was introduced with Classless Inter-Domain Routing (CIDR) and is called CIDR notation (RFC 4632). In IPv6 this is the only acceptable form to denote network or routing prefixes.

In classful networking in IPv4, prior to the introduction of CIDR, the network prefix could be directly obtained from the IP address, based on its highest order bit sequence. This determined the class (A, B, C) of the address and therefore the network mask. Since the introduction of CIDR, however, assignment of an IP address to a network interface requires two parameters, the address and its network mask.

In IPv4, on-link determination for an IP address is given simply by the address and netmask configuration, as the address cannot be disassociated from the on-link prefix.[2] For IPv6, however, on-link determination is different in detail and requires the Neighbor Discovery Protocol (NDP).[3] [4] IPv6 address assignment to an interface carries no requirement of a matching on-link prefix and vice versa, with the exception of link-local addresses.

While subnetting may improve network performance in an organizational network, it increases routing complexity, since each locally connected subnet must be represented by a separate entry in the routing tables of each connected router. However, by careful design of the network, routes to collections of more distant subnets within the branches of a tree-hierarchy can be aggregated by single routes. Variable-length subnet masking (VLSM) functionality in commercial routers made the introduction of CIDR seamless across the Internet and in enterprise networks.

IPv4 subnetting

The process of subnetting involves the separation of the network and subnet portion of an address from the host identifier. This is performed by a bitwise AND operation between the IP address and the (sub)network prefix. The result yields the network address or prefix, and the remainder is the host identifier.

Determining the network prefix

An IPv4 network mask consists of 32 bits, a sequence of ones (1) followed by a block of 0s. The trailing block of zeros (0) designates that part as being the host identifier.

The following example shows the separation of the network prefix and the host identifier from an address (192.168.5.130) and its associated /24 network mask (255.255.255.0). The operation is visualized in a table using binary address formats.

	Binary form	Dot-decimal notation
IP address	11000000.10101000.00000101.10000010	192.168.5.130
Subnet mask	11111111.11111111.11111111.00000000	255.255.255.0
Network prefix	11000000.10101000.00000101.00000000	192.168.5.0
Host part	00000000.00000000.00000000.10000010	0.0.0.130

The mathematical operation for calculating the network prefix is the binary and. The result of the operation yields the network prefix 192.168.5.0 and the host number 130.

Subnetting

Subnetting is the process of designating some high-order bits from the host part and grouping them with the network mask to form the **subnet mask**. This divides a network into smaller subnets. The following diagram modifies the example by moving 2 bits from the host part to the subnet mask to form a smaller subnet one fourth the previous size:

	Binary form	Dot-decimal notation
IP address	11000000.10101000.00000101.10000010	192.168.5.130
Subnet mask	11111111.11111111.11111111.**11**000000	255.255.255.192
Network prefix	11000000.10101000.00000101.10000000	192.168.5.128
Host part	00000000.00000000.00000000.00000010	0.0.0.2

Special addresses and subnets

Internet Protocol version 4 uses specially designated address formats to facilitate recognition of special address functionality. The first and the last subnets obtained by subnetting have traditionally had a special designation and, early on, special usage implications.[5] In addition, IPv4 uses the *all ones* host address, i.e. the last address within a network, for broadcast transmission to all hosts on the link.

Subnet zero and the all-ones subnet

The first subnet obtained from subnetting has all bits in the subnet bit group set to zero (0). It is therefore called *subnet zero*.[6] The last subnet obtained from subnetting has all bits in the subnet bit group set to one (1). It is therefore called the *all-ones subnet*.[7]

The IETF originally discouraged the production use of these two subnets at one point due to possible confusion of having a network and subnet with the same address.[8] The practice of avoiding subnet zero and the all-ones subnet was declared obsolete in 1995 by RFC 1878, an informational, but now historical RFC.[9]

Subnet and host counts

The number of subnetworks available, and the number of possible hosts in a network may be readily calculated. In the example (above) two bits were borrowed to create subnetworks, thus creating 4 (2^2) possible subnets.

Network	Network (binary)	Broadcast address
192.168.5.0/26	11000000.10101000.00000101.**00**000000	192.168.5.63
192.168.5.64/26	11000000.10101000.00000101.**01**000000	192.168.5.127
192.168.5.128/26	11000000.10101000.00000101.**10**000000	192.168.5.191
192.168.5.192/26	11000000.10101000.00000101.**11**000000	192.168.5.255

The RFC 950 specification reserves the subnet values consisting of all zeros (see above) and all ones (broadcast), reducing the number of available subnets by two. However, due to the inefficiencies introduced by this convention it was abandoned for use on the public Internet, and is only relevant when dealing with legacy equipment that does not implement CIDR. The only reason not to use the all-zeroes subnet is that it is ambiguous when the prefix length is not available. All CIDR-compliant routing protocols transmit both length and suffix. RFC 1878 provides a subnetting table with examples.

The remaining bits after the subnet are used for addressing hosts within the subnet. In the above example the subnet mask consists of 26 bits, leaving 6 bits for the host identifier. This allows for 64 combinations (2^6), however the all zeros value and all ones value are reserved for the network ID and broadcast address respectively, leaving 62 addresses.

In general the number of available hosts on a subnet is 2^n-2, where n is the number of bits used for the host portion of the address.

RFC 3021 specifies an exception to this rule when dealing with 31-bit subnet masks (i.e. 1-bit host identifiers). In such networks, usually point-to-point links, only two hosts (the end points) may be connected and a specification of network and broadcast addresses is not necessary.

A /24 network may be divided into the following subnets by increasing the subnet mask successively by one bit. This affects the total number of hosts that can be addressed in the /24 network (last column).

CIDR notation	Network mask	Available subnets	Usable hosts per subnet	Total usable hosts
/24	255.255.255.0	1	254	254
/25	255.255.255.128	2	126	252
/26	255.255.255.192	4	62	248
/27	255.255.255.224	8	30	240
/28	255.255.255.240	16	14	224
/29	255.255.255.248	32	6	192
/30	255.255.255.252	64	2	128
/31	255.255.255.254	128	2 *	256

**only applicable for point-to-point links*

IPv6 subnetting

The design of the IPv6 address space differs significantly from IPv4. The primary reason for subnetting in IPv4 is to improve efficiency in the utilization of the relatively small address space available, particularly to enterprises. No such limitations exist in IPv6, as the address space available, even to end-users, is large.

An RFC 4291 compliant subnet always uses IPv6 addresses with 64 bits for the host portion.[10] It therefore has a /64 routing prefix (128−64 = the 64 most significant bits). Although it is technically possible to use smaller subnets,[11] they are impractical for local area networks based on Ethernet technology, because 64 bits are required for stateless address auto configuration.[12] The Internet Engineering Task Force recommends the use of /64 subnets even for point-to-point links, which consist of only two hosts.[13]

IPv6 does not implement special address formats for broadcast traffic or network numbers,[14] and thus all addresses in a subnet are valid host addresses. The all-zeroes address is reserved as the Subnet-Router anycast address.[15]

The recommended allocation for an IPv6 customer site is an address space with an 80-bit (/48) prefix.[16] This provides 65536 subnets for a site. Despite this recommendation, other common allocations are /56 (72 bits) as well as /64 prefixes for a residential customer network.

Subnetting in IPv6 is based on the concepts of variable length subnet masking (VLSM) and the Classless Inter-Domain Routing methodology. It is used to route traffic between the global allocation spaces and within customer networks between subnets and the larger internet.

References

[1] RFC 950, *Internet Standard Subnetting Procedure*, J. Mogul, J. Postel (August 1985), page 1, 16

[2] RFC 1122, *Requirements for Internet Hosts -- Communication Layers*, Section 3.3.1, R. Braden, IETF (October 1989)

[3] RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, T. Narten et al. (September 2007)

[4] RFC 5942, IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes*, H. Singh, W. Beebee, E. Nordmark (July 2010)*

[5] "Document ID 13711 - Subnet Zero and the All-Ones Subnet" (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080093f18.shtml). Cisco Systems, Inc.. 2005-08-10. . Retrieved 2010-04-25. "Traditionally, it was strongly recommended that subnet zero and the all-ones subnet not be used for addressing. [...] Today, the use of subnet zero and the all-ones subnet is generally accepted and most vendors support their use."

[6] "Document ID 13711 - Subnet Zero and the All-Ones Subnet" (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080093f18.shtml). Cisco Systems, Inc.. 2005-08-10. . Retrieved 2010-04-23. "the first [...] subnet[...], known as subnet zero"

[7] "Document ID 13711 - Subnet Zero and the All-Ones Subnet" (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080093f18.shtml). Cisco Systems, Inc.. 2005-08-10. . Retrieved 2010-04-23. "[...] the last subnet[...], known as [...] the all-ones subnet"

[8] Jeffrey Mogul; Jon Postel (August 1985). [RFC 950 "Internet Standard Subnetting Procedure"]. Internet Engineering Task Force (IETF). p. 6. RFC 950. Retrieved 2010-04-23. "It is useful to preserve and extend the interpretation of these special addresses in subnetted networks. This means the values of all zeros and all ones in the subnet field should not be assigned to actual (physical) subnets."

[9] Troy Pummill; Bill Manning (December 1995). [RFC 1878 "Variable Length Subnet Table For IPv4"]. RFC 1878. "This practice is obsolete! Modern software will be able to utilize all definable networks." Note: RFC 1878 is an Informational RFC, and has been demoted to category *Historic*.

[10] "IP Version 6 Addressing Architecture - section 2.5.1. Interface Identifiers" (http://tools.ietf.org/html/rfc4291#section-2.5.1). Internet Engineering Task Force. . Retrieved 2011-02-13. "For all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long and to be constructed in Modified EUI-64 format."

[11] "IPv6 Stateless Address Autoconfiguration - section 5.5.3.(d) Router Advertisement Processing" (http://tools.ietf.org/html/rfc4862#section-5.5.3). Internet Engineering Task Force. . Retrieved 2011-02-13. "It is the responsibility of the system administrator to ensure that the lengths of prefixes contained in Router Advertisements are consistent with the length of interface identifiers for that link type. [...] an implementation should not assume a particular constant. Rather, it should expect any lengths of interface identifiers."

[12] "Transmission of IPv6 Packets over Ethernet Networks - section 4 Stateless Autoconfiguration" (http://tools.ietf.org/html/rfc2464#section-4). Internet Engineering Task Force. . "The Interface Identifier [AARCH] for an Ethernet interface is based on the EUI-64 identifier [EUI64] derived from the interface's built-in 48-bit IEEE 802 address. [...] An IPv6 address prefix used for stateless autoconfiguration [ACONF] of an Ethernet interface must have a length of 64 bits."

[13] "Use of /127 Prefix Length Between Routers Considered Harmful" (http://tools.ietf.org/html/rfc3627#section-4). Internet Engineering Task Force. . "One could use /64 for subnets, including point-to-point links. [...] Failing that, /126 does not have this problem, and it can be

used safely on a point-to-point link"

[14] "IP Version 6 Addressing Architecture - section 2 IPv6 Addressing" (http://tools.ietf.org/html/rfc4291#section-2). Internet Engineering Task Force. . "There are no broadcast addresses in IPv6, their function being superseded by multicast addresses. [...] In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded."

[15] "IP Version 6 Addressing Architecture - section 2.6.1 Required Anycast Address" (http://tools.ietf.org/html/rfc4291#section-2.6.1). Internet Engineering Task Force. . "This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero."

[16] "IPv6 Addressing Plans" (http://www.getipv6.info/index.php?title=IPv6_Addressing_Plans&oldid=2998). ARIN IPv6 Wiki. . Retrieved 2010-04-25. "All customers get one /48 unless they can show that they need more than 65k subnets. [...] If you have lots of consumer customers you may want to assign /56s to private residence sites."

Further reading

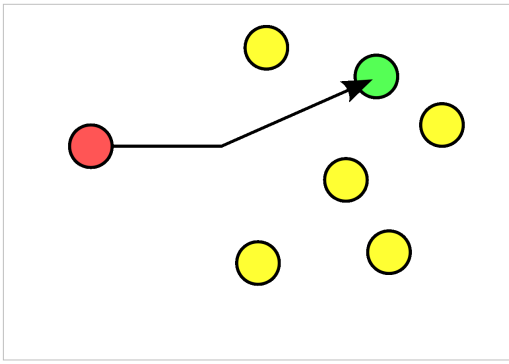
- RFC 1812 Requirements for IPv4 Routers
- RFC 917 Utility of subnets of Internet networks
- RFC 1101 DNS Encodings of Network Names and Other Type
- Blank, Andrew G. *TCP/IP Foundations Technology Fundamentals for IT Success*. San Francisco, London: Sybex, Copyright 2004.
- Lammle, Todd. *CCNA Cisco Certified Network Associate Study Guide 5th Edition*. San Francisco, London: Sybex, Copyright 2005.
- Groth, David and Toby Skandier. *Network + Study Guide*, 4th Edition. San Francisco, London: Wiley Publishing, Inc., Copyright 2005.

External links

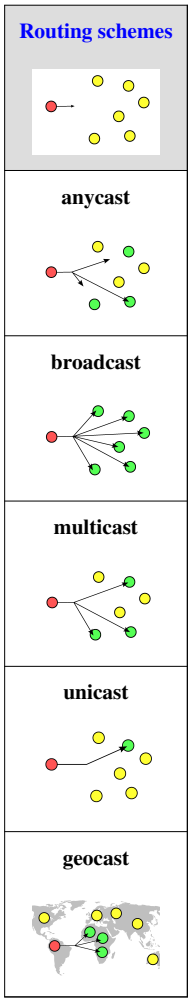
- IP Address Subnetting Tutorial (http://www.ralphb.net/IPSubnet/)
- Cisco-IP Addressing and Subnetting for New Users (http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a00800a67f5.shtml)
- IP subnetting made easy (http://techrepublic.com.com/5100-1035_11-6089187.html?tag=fdpop)
- Subnetworking (http://www.dmoz.org/Computers/Internet/Protocols/IP/Addressing//) at the Open Directory Project

Unicast

In computer networking, **unicast** transmission is the sending of messages to a single network destination identified by a unique address[1] .



Addressing methodologies



The term *unicast* is contrasted with the term *broadcast* which means transmitting the same data to all possible destinations. Another multi-destination distribution method, multicasting, sends data only to *interested* destinations by using special address assignments.

Unicast messaging is used for all network processes in which a private or unique resource is requested.

Certain network applications which are mass-distributed are too costly to be conducted with unicast transmission since each network connection consumes computing resources on the sending host and requires its own separate network bandwidth for transmission. Such applications include streaming media of many forms. Internet radio stations using unicast connections may have high bandwidth costs.

These terms are also used by streaming content providers' services. Unicast-based media servers open and provide a stream for each unique user. Multicast-based servers can support a larger audience by serving content simultaneously to multiple users.

References

[1] http://www.science.uva.nl/research/air/projects/old_projects/ipv6/IPv6_uni.htm A unicast address is an identifier for an interface. University of Amsterdam

External links

- "Differences Between Multicast and Unicast" (http://support.microsoft.com/kb/291786). Microsoft. Retrieved 2008-02-04.
- "What Is Unicast IPv4 Routing?" (http://technet.microsoft.com/de-de/library/cc736574(WS.10).aspx). Microsoft. Retrieved 2010-09-30.

Transport Layer

Additive increase/multiplicative decrease

The **additive increase/multiplicative-decrease** (**AIMD**) algorithm is a feedback control algorithm best known for its use in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when a congestion takes place. Multiple flows using AIMD congestion control will eventually converge to use equal amounts of a contended link.[1] The related schemes of multiplicative-increase/multiplicative-decrease (MIMD) and additive-increase/additive-decrease (AIAD) do not converge.

Algorithm

The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by a fixed amount every round trip time. When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half after loss. The result is a saw-tooth behavior that represents the probe for bandwidth.

AIMD requires a binary signal of congestion. Most frequently, packet loss serves as the signal; the multiplicative decrease is triggered when a timeout or acknowledgement message indicates a packet was lost. It is also possible for in-network mechanisms to mark congestion (without discarding packets) as in Explicit Congestion Notification (ECN).

Mathematical Formula

Let $w(t)$ be the sending rate (e.g. the congestion window) during time slot t, a ($a > 0$) be the additive increase factor, and b ($0 < b < 1$) be the multiplicative decrease factor.

$$w(t+1) = \begin{cases} w(t) + a & \text{if congestion is not detected} \\ w(t) \times b & \text{if congestion is detected} \end{cases}$$

In TCP, after slow start, the additive increase factor a is typically one MSS (maximum segment size) per round-trip time, and the multiplicative decrease factor b is typically 1/2.

Protocols

AIMD congestion avoidance is or was used in:

- Transmission Control Protocol (TCP)
- Stream Control Transmission Protocol (SCTP)
- OSI Transport Class 4[1]
- DCCP (in some modes)[2]
- DECnet[1]

References

[1] Chiu, Dah-Ming; Raj Jain (1989). "Analysis of increase and decrease algorithms for congestion avoidance in computer networks". *Computer Networks and ISDN systems* **17**: 1–14.

[2] Floyd. "4341: RFC Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control" (https://tools.ietf.org/html/rfc4341). .

Congestion window

In Transmission Control Protocol (TCP), the **congestion window** is one of the factors that determines the number of bytes that can be outstanding at any time. This is not to be confused with the TCP window size which is maintained by the receiver. This is a means of stopping the link between two places from getting overloaded with too much traffic. The size of this window is calculated by estimating how much congestion there is between the two places. The sender maintains the congestion window.

When a connection is set up, the congestion window is set to the maximum segment size (MSS) allowed on that connection. Further variance in the collision window is dictated by an Additive Increase/Multiplicative Decrease approach.

This means that if all segments are received and the acknowledgments reach the sender on time, some constant is added to the window size. The window keeps growing exponentially until a timeout occurs or the receiver reaches its limit (a threshold value "ssthresh"). After this the congestion window increases linearly at the rate of 1/(congestion window)packets on each new acknowledgement received.

On timeout:

1. Congestion window is reset to 1 MSS
2. "ssthresh" is set to half the window size before packet loss started
3. "slow start" is initiated.

A system administrator may adjust the maximum window size limit, or adjust the constant added during additive increase, as part of TCP tuning.

The flow of data over a TCP connection is also controlled by the use of the receiver advertised TCP Receive Window. By comparing its own congestion window with the receive window of the receiver, a sender can determine how much data it may send at any given time.

Connection-oriented protocol

A connection-oriented networking protocol is one that establishes a communication session, then delivers a stream of data in the same order as it was sent. It may be a circuit switched connection, or a virtual circuit connection in a packet switched network. In the latter case, it identifies traffic flows by some *connection identifier* rather than by explicitly listing source and destination addresses. Typically, this connection identifier is a small integer (10 bits for Frame Relay, 24 for ATM, for example). This makes network switches substantially faster (as routing tables are just simple look-up tables, and are trivial to implement in hardware). The impact is so great, in fact, that even characteristically connectionless protocols, such as IP traffic, are being *tagged* with connection-oriented header prefixes (e.g., as with MPLS, or IPv6's built-in Flow ID field). Example of a connection-oriented protocol at the transport layer is the TCP protocol.

Connection-oriented protocols are not necessarily reliable protocols. ATM and Frame Relay, for example, are both examples of a connection-oriented, unreliable protocol. There are also reliable connectionless protocols as well, such as AX.25 when it passes data in I-frames. But this combination is rare, and reliable-connectionless is uncommon in commercial and academic networks.

Connection-oriented protocols handle real-time traffic substantially more efficiently than connectionless protocols, which is why ATM has yet to be replaced with Ethernet for carrying real-time, isochronous traffic streams, especially in heavily aggregated networks like backbones, where the motto "bandwidth is cheap" fails to deliver on its promise. Experience has also shown that over-provisioning bandwidth does not resolve all quality of service issues. Hence, (10-)gigabit Ethernet is not expected to replace ATM at this time.

Some connection-oriented protocols have been designed or altered to accommodate both connection-oriented and connectionless data.[1]

List of connection-oriented protocols

- TCP
- SMTP
- DCCP
- Connection-oriented Ethernet
- ATM
- Frame Relay
- TIPC
- SCTP
- IPX/SPX

References

[1] Ramos-Escano et al. (2005-06-02). "US Patent Application Publication 2005/0117529 A1" (http://www.google.com/patents?id=hzKUAAAAEBAJ). . Retrieved 2008-05-19.

Connectionless protocol

In telecommunications, **connectionless** describes communication between two network end points in which a message can be sent from one end point to another without prior arrangement. The device at one end of the communication transmits data addressed to the other, without first ensuring that the recipient is available and ready to receive the data. Some protocols allow for error correction by requested retransmission. Internet Protocol (IP) and User Datagram Protocol (UDP) are connectionless protocols.

Connectionless protocols are also described as stateless because the endpoints have no protocol-defined way to remember where they are in a "conversation" of message exchanges.

List of connectionless protocols

- Hypertext Transfer Protocol
- IP (internet layer, can also be used for connections)
- UDP
- ICMP
- IPX
- TIPC
- NetBEUI

Fast retransmit

Fast Retransmit is an enhancement to TCP which reduces the time a sender waits before retransmitting a lost segment.

A TCP sender uses a timer to recognize lost segments. If an acknowledgement is not received for a particular segment within a specified time (a function of the estimated Round-trip delay time), the sender will assume the segment was lost in the network, and will retransmit the segment.

Duplicate acknowledgement is the basis for fast retransmit mechanism which works as follows: after receiving a packet (i.e., sequence number 1), the receiver sends an acknowledgement adding 1 with the sequence number (i.e., sequence number 2) which means that the receiver receives the packet number 1 and it expects packet number 2 from the sender. Lets assume that three subsequent packets have been lost. In the meantime the receiver receives the packet number 5 and 6. After receiving packet number 5, the receiver sends an acknowledgement with the sequence number 2 and 6. When the receiver receives packet number 6, it sends acknowledgement with the sequence number 2 and 7. In this way, the sender receives more than one acknowledgement with the same sequence number 2 which is called duplicate acknowledgement.

The fast retransmit enhancement works as follows: if a TCP sender receives a specified number of acknowledgements which is usually set to three duplicate acknowledgements with the same acknowledge number (that is, a total of four acknowledgements with the same acknowledgement number), the sender can be reasonably confident that the segment with the next higher sequence number was dropped, and will not arrive out of order. The sender will then retransmit the packet that was presumed dropped before waiting for its timeout.

External links

- Mark Allman, Vern Paxson, W. Richard Stevens (April 1999). "Fast Retransmit/Fast Recovery" [1]. *TCP Congestion Control* [2]. IETF. sec. 3.2. RFC 2581. Retrieved 2010-05-01.
- TCP Congestion Handling and Congestion Avoidance Algorithms [3] — The TCP/IP Guide

References

[1] http://tools.ietf.org/html/rfc2581#section-3.2
[2] http://tools.ietf.org/html/rfc2581
[3] http://www.tcpipguide.com/free/t_TCPCongestionHandlingandCongestionAvoidanceAlgorit-3.htm

Karn's Algorithm

Karn's Algorithm addresses the problem of getting accurate estimates of the round-trip time for messages when using TCP. The algorithm was proposed by Phil Karn in 1987.[1]

Accurate round trip estimates in TCP can be difficult to calculate because of an ambiguity created by retransmitted segments. The round trip time is estimated as the difference between the time that a segment was sent and the time that its acknowledgment was returned to the sender, but when packets are re-transmitted there is an ambiguity: the acknowledgment may be a response to the first transmission of the segment or to a subsequent re-transmission.

Karn's Algorithm ignores retransmitted segments when updating the round trip time estimate. Round trip time estimation is based only on unambiguous acknowledgments, which are acknowledgments for segments that were sent only once.

This simplistic implementation of Karn's algorithm can lead to problems as well. Consider what happens when TCP sends a segment after a sharp increase in delay. Using the prior round trip time estimate, TCP computes a timeout and retransmits a segment. If TCP never acknowledges retransmitted packets, the round trip estimate time will never be updated, and TCP will continue retransmitting segments without adjusting to the increased delay.

A solution to this problem is to incorporate transmission timeouts with a timer backoff strategy. The timer backoff strategy computes an initial timeout. If the timer expires and causes a retransmission, TCP increases the timeout generally by a factor of 2.

new_timeout = 2 * timeout

This algorithm has proven to be extremely effective in networks with high packet loss. [2]

External links

- RFC 2581 - TCP Congestion Control
- RFC 2988 - Computing TCP's Retransmission Timer

References

[1] Karn, Phil; Craig Partridge (11-13 August). "Improving Round-Trip Time Estimates in Reliable Transport Protocols" (http://www.ka9q.net/papers/rtt.ps.gz) (PS). *ACM SIGCOMM '87*. pp. 2–7. . Retrieved 2009-07-15.
[2] Comer, Douglas E. (2006). *Internetworking with TCP/IP* (5 ed.). Prentice Hall: Upper Saddle River, NJ.

Maximum segment size

The **maximum segment size** (**MSS**) is a parameter of the TCP protocol that specifies the largest amount of data, specified in octets, that a computer or communications device can receive in a single TCP segment, and therefore in a single IP datagram. It does not count the TCP header or the IP header.[1] The IP datagram containing a TCP segment may be self-contained within a single packet, or it may be reconstructed from several fragmented pieces; either way, the MSS limit applies to the total amount of data contained within the final reconstructed TCP segment.

Therefore: Headers + MSS ≤ MTU

The Minimum MSS = Maximum datagram size - IP header size - TCP header size
So every IPv4 host is required to be able to handle an MSS of at least 536 octets (= 576 - 20 - 20)
and every IPv6 host is required to be able to handle an MSS of at least 1220 octets (= 1280 - 40 - 20).[2]

For most computer users, the MSS option is established by operating system on the SYN packet during the TCP handshake. Each direction of data flow can use a different MSS.

Further reading

- Comer, Douglas E. (2006). *Internetworking with TCP/IP*. **Vol. 1** (5/E ed.). Upper Saddle River, NJ, USA: Prentice Hall.
- Kozierok, Charles M. (2005-09-20). *The TCP/IP Guide* [3] (3.0 ed.). Retrieved 2011-09-08.

References

[1] RFC 879 (http://tools.ietf.org/html/rfc879#section-3), page 2, Section 3, "The MSS counts only data octets in the segment, it does not count the TCP header or the IP header."

[2] RFC 2460 (http://tools.ietf.org/html/rfc2460#section-8.3), page 28, Section 8.3

[3] http://www.tcpipguide.com/free/t_TCPMaximumSegmentSizeMSSandRelationshiptoIPDatagra-2.htm

Nagle's algorithm

Nagle's algorithm, named after John Nagle, is a means of improving the efficiency of TCP/IP networks by reducing the number of packets that need to be sent over the network.

Nagle's document, *Congestion Control in IP/TCP Internetworks* (RFC 896) describes what he called the 'small packet problem', where an application repeatedly emits data in small chunks, frequently only 1 byte in size. Since TCP packets have a 40 byte header (20 bytes for TCP, 20 bytes for IPv4), this results in a 41 byte packet for 1 byte of useful information, a huge overhead. This situation often occurs in Telnet sessions, where most keypresses generate a single byte of data that is transmitted immediately. Worse, over slow links, many such packets can be in transit at the same time, potentially leading to congestion collapse.

Nagle's algorithm works by combining a number of small outgoing messages, and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, so that output can be sent all at once.

Algorithm

```
if there is new data to send
  if the window size >= MSS and available data is >= MSS
    send complete MSS segment now
  else
    if there is unconfirmed data still in the pipe
      enqueue data in the buffer until an acknowledge is received
    else
      send data immediately
    end if
  end if
end if
```

where *MSS = maximum segment size*.

This algorithm interacts badly with TCP delayed acknowledgments, a feature introduced into TCP at roughly the same time in the early 1980s, but by a different group. With both algorithms enabled, applications that do two successive writes to a TCP connection, followed by a read that will not be fulfilled until after the data from the second write has reached the destination, experience a constant delay of up to 500 milliseconds, the "ACK delay". For this reason, TCP implementations usually provide applications with an interface to disable the Nagle algorithm. This is typically called the `TCP_NODELAY` option. It is, however, not recommended to disable this. A solution, recommended by Nagle himself, is to keep from sending small single writes and buffer up application writes then send (or with the help of writev() call).

> "The user-level solution is to avoid write-write-read sequences on sockets. write-read-write-read is fine. write-write-write is fine. But write-write-read is a killer. So, if you can, buffer up your little writes to TCP and send them all at once. Using the standard UNIX I/O package and flushing write before each read usually works."[1]

Thus, you leave the Nagle algorithm on, but basically don't use it because you call flush on your writes.

The tinygram problem and silly window syndrome are sometimes confused. The tinygram problem occurs when the window is almost empty. Silly window syndrome occurs when the window is almost full.

Interactions with real-time systems

Applications that expect real time responses can react poorly with Nagle's algorithm. Applications such as networked multiplayer video games expect that actions in the game are sent immediately, while the algorithm purposefully delays transmission, increasing bandwidth at the expense of latency. For this reason applications with low-bandwidth time-sensitive transmissions typically use `TCP_NODELAY` to bypass the Nagle delay.[2]

References

[1] Boosting Socket Performance on Linux - Slashdot (http://developers.slashdot.org/comments.pl?sid=174457&threshold=1&commentsort=0&mode=thread&cid=14515105)

[2] Bug 17868 – Some Java applications are slow on remote X connections (https://bugs.freedesktop.org/show_bug.cgi?id=17868)

- Larry L. Peterson, Bruce S. Davie (2007). *Computer Networks: A Systems Approach* (http://books.google.com/books?id=fknMX18T40cC&pg=PA402&dq=Nagle's+algorithm#PPA402,M1) (4 ed.). Morgan Kaufmann. p. 402–403. ISBN 0123740134.

External links

- Nagle's algorithm (http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci754347,00.html)
- TCP Performance problems caused by interaction between Nagle's Algorithm and Delayed ACK (http://www.stuartcheshire.org/papers/NagleDelayedAck/)
- Design issues - Sending small data segments over TCP with Winsock (http://support.microsoft.com/kb/214397/en-us)

Negative-acknowledge character

- In telecommunications, a **negative-acknowledge character** (NAK or NACK) is a transmission control character sent by a station as a negative response to the station with which the connection has been set up.
- In Binary Synchronous Communications protocol, the NAK is used to indicate that an error was detected in the previously received block and that the receiver is ready to accept retransmission of that block.
- In multipoint systems, the NAK is used as the not-ready reply to a poll.
- In the ASCII code, the NAK character is 0x15 (hexadecimal), or `^U`.

References

- *This article incorporates public domain material from the General Services Administration* document "Federal Standard 1037C" [1] (in support of MIL-STD-188).

Network congestion

In data networking and queueing theory, **network congestion** occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queueing delay, packet loss or the blocking of new connections. A consequence of these latter two is that incremental increases in offered load lead either only to small increases in network throughput, or to an actual reduction in network throughput.

Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion even after the initial load has been reduced to a level which would not normally have induced network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as **congestive collapse**.

Modern networks use congestion control and network congestion avoidance techniques to try to avoid congestion collapse. These include: exponential backoff in protocols such as 802.11's CSMA/CA and the original Ethernet, window reduction in TCP, and fair queueing in devices such as routers. Another method to avoid the negative effects of network congestion is implementing priority schemes, so that some packets are transmitted with higher priority than others. Priority schemes do not solve network congestion by themselves, but they help to alleviate the effects of congestion for some services. An example of this is 802.1p. A third method to avoid network congestion is the explicit allocation of network resources to specific flows. One example of this is the use of Contention-Free Transmission Opportunities (CFTXOPs) in the ITU-T G.hn standard, which provides high-speed (up to 1 Gbit/s) Local area networking over existing home wires (power lines, phone lines and coaxial cables).

RFC 2914 addresses the subject of congestion control in detail.

Network capacity

The fundamental problem is that all network resources are limited, including router processing time and link throughput.

However:

- today's (2006) Wireless LAN effective bandwidth throughput (15-100Mbit/s) is easily filled by a single personal computer.
- Even on fast computer networks (e.g. 1 Gbit), the backbone can easily be congested by a few servers and client PCs.
- Because P2P scales very well, file transmissions by P2P have no problem filling and will fill an uplink or some other network bottleneck, particularly when nearby peers are preferred over distant peers.
- Denial of service attacks by botnets are capable of filling even the largest Internet backbone network links (40 Gbit/s as of 2007), generating large-scale network congestion

Congestive collapse

Congestive collapse (or **congestion collapse**) is a condition which a packet switched computer network can reach, when little or no useful communication is happening due to congestion. Congestion collapse generally occurs at choke points in the network, where the total incoming traffic to a node exceeds the outgoing bandwidth. Connection points between a local area network and a wide area network are the most likely choke points.

When a network is in such a condition, it has settled (under overload) into a stable state where traffic demand is high but little useful throughput is available, and there are high levels of packet delay and loss (caused by routers discarding packets because their output queues are too full) and general quality of service is extremely poor.

History

Congestion collapse was identified as a possible problem as far back as 1984 (RFC 896, dated 6 January). It was first observed on the early Internet in October 1986, when the NSFnet phase-I backbone dropped three orders of magnitude from its capacity of 32 kbit/s to 40 bit/s, and this continued to occur until end nodes started implementing Van Jacobson's congestion control between 1987 and 1988.

Cause

When more packets were sent than could be handled by intermediate routers, the intermediate routers discarded many packets, expecting the end points of the network to retransmit the information. However, early TCP implementations had very bad retransmission behavior. When this packet loss occurred, the end points sent *extra* packets that repeated the information lost; doubling the data rate sent, exactly the opposite of what should be done during congestion. This pushed the entire network into a 'congestion collapse' where most packets were lost and the resultant throughput was negligible.

Congestion control

Congestion control concerns controlling traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. It should not be confused with flow control, which prevents the sender from overwhelming the receiver.

Theory of congestion control

The modern theory of congestion control was pioneered by Frank Kelly, who applied microeconomic theory and convex optimization theory to describe how individuals controlling their own rates can interact to achieve an "optimal" network-wide rate allocation.

Examples of "optimal" rate allocation are max-min fair allocation and Kelly's suggestion of proportional fair allocation, although many others are possible.

The mathematical expression for optimal rate allocation is as follows. Let x_i be the rate of flow i, C_l be the capacity of link l, and r_{li} be 1 if flow i uses link l and 0 otherwise. Let x, c and R be the corresponding vectors and matrix. Let $U(x)$ be an increasing, strictly convex function, called the utility, which measures how much benefit a user obtains by transmitting at rate x. The optimal rate allocation then satisfies

$$\max_x \sum_i U(x_i)$$

such that $Rx \leq c$

The Lagrange dual of this problem decouples, so that each flow sets its own rate, based only on a "price" signalled by the network. Each link capacity imposes a constraint, which gives rise to a Lagrange multiplier, p_l. The sum of these Lagrange multipliers, $y_i = \sum_l p_l r_{li}$, is the price to which the flow responds.

Congestion control then becomes a distributed optimisation algorithm for solving the above problem. Many current congestion control algorithms can be modelled in this framework, with p_l being either the loss probability or the queueing delay at link l.

A major weakness of this model is that it assumes all flows observe the same price, while sliding window flow control causes "burstiness" which causes different flows to observe different loss or delay at a given link.

Classification of congestion control algorithms

There are many ways to classify congestion control algorithms:

- By the type and amount of feedback received from the network: Loss; delay; single-bit or multi-bit explicit signals
- By incremental deployability on the current Internet: Only sender needs modification; sender and receiver need modification; only router needs modification; sender, receiver and routers need modification.
- By the aspect of performance it aims to improve: high bandwidth-delay product networks; lossy links; fairness; advantage to short flows; variable-rate links
- By the fairness criterion it uses: max-min, proportional, "minimum potential delay"

Avoidance

The prevention of network congestion and collapse requires two major components:

1. A mechanism in routers to reorder or drop packets under overload,
2. End-to-end flow control mechanisms designed into the end points which respond to congestion and behave appropriately.

The correct end point behaviour is usually still to repeat dropped information, but progressively slow the rate that information is repeated. Provided all end points do this, the congestion lifts and good use of the network occurs, and the end points all get a fair share of the available bandwidth. Other strategies such as slow-start ensure that new connections don't overwhelm the router before the congestion detection can kick in.

The most common router mechanisms used to prevent congestive collapses are fair queueing and other scheduling algorithms, and random early detection, or RED, where packets are randomly dropped proactively triggering the end points to slow transmission before congestion collapse actually occurs. Fair queueing is most useful in routers at choke points with a small number of connections passing through them. Larger routers must rely on RED.

Some end-to-end protocols are better behaved under congested conditions than others. TCP is perhaps the best behaved. The first TCP implementations to handle congestion well were developed in 1984, but it was not until Van Jacobson's inclusion of an open source solution in the Berkeley Standard Distribution UNIX ("BSD") in 1988 that good TCP implementations became widespread.

UDP does not, in itself, have any congestion control mechanism. Protocols built atop UDP must handle congestion in their own way. Protocols atop UDP which transmit at a fixed rate, independent of congestion, can be troublesome. Real-time streaming protocols, including many Voice over IP protocols, have this property. Thus, special measures, such as quality-of-service routing, must be taken to keep packets from being dropped from streams.

In general, congestion in pure datagram networks must be kept out at the periphery of the network, where the mechanisms described above can handle it. Congestion in the Internet backbone is very difficult to deal with. Fortunately, cheap fiber-optic lines have reduced costs in the Internet backbone. The backbone can thus be provisioned with enough bandwidth to keep congestion at the periphery.

Practical network congestion avoidance

Implementations of connection-oriented protocols, such as the widely-used TCP protocol, generally watch for packet errors, losses, or delays (see Quality of Service) in order to adjust the transmit speed. There are many different network congestion avoidance processes, since there are a number of different trade-offs available. [1]

TCP/IP congestion avoidance

The TCP congestion avoidance algorithm is the primary basis for congestion control in the Internet. [2] [3] [4] [5] [6]

Problems occur when many concurrent TCP flows are experiencing port queue buffer tail-drops. Then TCP's automatic congestion avoidance is not enough. All flows that experience port queue buffer tail-drop will begin a TCP retrain at the same moment - this is called TCP global synchronization.

Active Queue Management (AQM)

Purpose

"Recommendations on Queue Management and Congestion Avoidance in the Internet" (RFC 2309[7]) states that:

- Fewer packets will be dropped with Active Queue Management (AQM).
- The link utilization will increase because less TCP global synchronization will occur.
- By keeping the average queue size small, queue management will reduce the delays and jitter seen by flows.
- The connection bandwidth will be more equally shared among connection oriented flows, even without flow-based RED or WRED.

Random early detection

One solution is to use random early detection (RED) on network equipments port queue buffer. [8] [9] On network equipment ports with more than one queue buffer, weighted random early detection (WRED) could be used if available.

RED indirectly signals to sender and receiver by deleting some packets, e.g. when the average queue buffer lengths are more than e.g. 50% (lower threshold) filled and deletes linearly more or (better according to paper) cubical more packets, [10] up to e.g. 100% (higher threshold). The average queue buffer lengths are computed over 1 second at a time.

Robust random early detection (RRED)

Robust Random Early Detection (RRED) algorithm was proposed to improve the TCP throughput against Denial-of-Service (DoS) attacks, particularly Low-rate Deinal-of-Service (LDoS) attacks. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under Low-rate Denial-of-Service [11] (LDoS) attacks due to the oscillating TCP queue size caused by the attacks [12] . RRED algorithm can significantly improve the performance of TCP under Low-rate Denial-of-Service attacks [12] .

Recent Publications in low-rate Denial-of-Service (DoS) attacks [13]

Flowbased-RED/WRED

Some network equipment are equipped with ports that can follow and measure each flow (**flowbased-RED/WRED**) and are hereby able to signal to a too big bandwidth flow according to some QoS policy. A policy could divide the bandwidth among all flows by some criteria.

IP ECN

Another approach is to use IP ECN.[14] ECN is only used when the two hosts signal that they want to use it. With this method, an ECN bit is used to signal that there is explicit congestion. This is better than the indirect packet delete congestion notification performed by the RED/WRED algorithms, but it requires explicit support by both hosts to be effective. [15] Some outdated or buggy network equipment drops packets with the ECN bit set, rather than ignoring the bit. More information on the status of ECN including the version required for Cisco IOS, by Sally Floyd,[8] one of the authors of ECN.

When a router receives a packet marked as ECN capable and anticipates (using RED) congestion, it will set an ECN-flag notifying the sender of congestion. The sender then ought to decrease its transmission bandwidth; e.g. by decreasing the tcp window size (sending rate) or by other means.

Cisco AQM: Dynamic buffer limiting (DBL)

Cisco has taken a step further in their Catalyst 4000 series with engine IV and V. Engine IV and V has the possibility to classify all flows in "aggressive" (bad) and "adaptive" (good). It ensures that no flows fill the port queues for a long time. **DBL** can utilize **IP ECN** instead of packet-delete-signalling. [16] [17]

TCP Window Shaping

Congestion avoidance can also efficiently be achieved by reducing the amount of traffic flowing into a network. When an application requests a large file, graphic or web page, it usually advertises a "window" of between 32K and 64K. This results in the server sending a full window of data (assuming the file is larger than the window). When there are many applications simultaneously requesting downloads, this data creates a congestion point at an upstream provider by flooding the queue much faster than it can be emptied. By using a device to reduce the window advertisement, the remote servers will send less data, thus reducing the congestion and allowing traffic to flow more freely. This technique can reduce congestion in a network by a factor of 40.

Side effects of congestive collapse avoidance

Radio links

The protocols that avoid congestive collapse are often based on the idea that data loss on the Internet is caused by congestion. This is true in nearly all cases; errors during transmission are rare on today's fiber based Internet. However, this causes WiFi, 3G or other networks with a radio layer to have poor throughput in some cases since wireless networks are susceptible to data loss due to interference. The TCP connections running over a radio based physical layer see the data loss and tend to believe that congestion is occurring when it isn't and erroneously reduce the data rate sent.

Short-lived connections

The slow-start protocol performs badly for short-lived connections. Older web browsers would create many consecutive short-lived connections to the web server, and would open and close the connection for each file requested. This kept most connections in the slow start mode, which resulted in poor response time.

To avoid this problem, modern browsers either open multiple connections simultaneously or reuse one connection for all files requested from a particular web server. However, the initial performance can be poor, and many connections never get out of the slow-start regime, significantly increasing latency.

References

[1] TCP Tunnels: Avoiding Congestion Collapse (2000) (http://citeseer.ist.psu.edu/lee00tcp.html)

[2] Van Jacobson, Michael J. Karels. Congestion Avoidance and Control (http://citeseer.ist.psu.edu/484335.html) (1988). *Proceedings of the Sigcomm '88 Symposium*, vol.18(4): pp.314–329. Stanford, CA. August, 1988. This paper originated many of the congestion avoidance algorithms used in TCP/IP.

[3] RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

[4] RFC 2581 - TCP Congestion Control

[5] RFC 3390 - TCP Increasing TCP's Initial Window

[6] TCP Congestion Avoidance Explained via a Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/TCP_Congestion_Avoidance.pdf)

[7] RFC 2309 - April 1998: Recommendations on Queue Management and Congestion Avoidance in the Internet

[8] Sally Floyd: RED (Random Early Detection) Queue Management (http://www.icir.org/floyd/red.html)

[9] Sally Floyd, Van Jacobson. Random Early Detection Gateways for Congestion Avoidance (http://citeseer.ist.psu.edu/462978.html) (1993). *IEEE/ACM Transactions on Networking*, vol.1(4): pp.397–413. Invented Random Early Detection (RED) gateways.

[10] An Analytical RED Function Design Guaranteeing Stable System Behavior (http://www.ist-mobydick.org/publications/aqm_iscc2003.pdf) Quote: "...The advantage of this function lies not only in avoiding heavy oscillations but also in avoiding link under-utilization at low loads. The applicability of the derived function is independent of the load range, no parameters are to be adjusted. Compared to the original linear drop function applicability is extended by far...Our example with realistic system parameters gives an approximation function of the cubic of the queue size..."

[11] http://sites.google.com/site/cwzhangres/home/files/RREDRobustREDAlgorithmtoCounterLow-rateDenial-of-ServiceAttacks.pdf?attredirects=0

[12] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen, RRED: Robust RED Algorithm to Counter Low-rate Denial-of-Service Attacks (http://sites.google.com/site/cwzhangres/home/files/RREDRobustREDAlgorithmtoCounterLow-rateDenial-of-ServiceAttacks.pdf?attredirects=0), IEEE Communications Letters, vol. 14, pp. 489-491, 2010. Ref (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5456075)

[13] http://sites.google.com/site/cwzhangres/home/posts/recentpublicationsinlow-ratedosattacks

[14] RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP

[15] Comparative study of RED, ECN and TCP Rate Control (1999) (http://citeseer.ist.psu.edu/bagal99comparative.html)

[16] Active Queue Management (http://www.cisco.com/univercd/cc/td/doc/product/lan/cat4000/12_1_19/config/qos.htm#1271743)

[17] Enabling Dynamic Buffer Limiting (http://www.cisco.com/univercd/cc/td/doc/product/lan/cat4000/12_1_19/config/qos.htm#1271759)

- "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice" by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, ISBN 0-12-370549-5)
- RFC 2914 (http://tools.ietf.org/html/rfc2914) - Congestion Control Principles, Sally Floyd, September, 2000
- RFC 896 (http://tools.ietf.org/html/rfc896) - "Congestion Control in IP/TCP", John Nagle, 6 January 1984
- Introduction to *Congestion Avoidance and Control* (http://ee.lbl.gov/papers/congavoid.pdf), Van Jacobson and Michael J. Karels, November, 1988

Books

- "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice" by John Evans, Clarence Filsfils (Morgan Kaufmann, 2007, ISBN 0-12-370549-5)

External links

- Nagle, J. RFC 896: *Congestion control in IP/TCP internetworks* (1984)
- Floyd, S. RFC 2914: *Congestion control principles* (2000)
- Floyd, S. and K. Fall, *Promoting the Use of End-to-End Congestion Control in the Internet* (http://www.aciri.org/floyd/end2end-paper.html) (IEEE/ACM Transactions on Networking, August 1999)
- Sally Floyd, *On the Evolution of End-to-end Congestion Control in the Internet: An Idiosyncratic View* (http://www.ima.umn.edu/talks/workshops/10-22-24.99/floyd/floyd.pdf) (IMA Workshop on Scaling Phenomena in Communication Networks, October 1999) (*pdf format*)
- Linktionary term: Queuing (http://www.linktionary.com/q/queuing.html)
- Pierre-Francois Quet, Sriram Chellappan, Arjan Durresi, Mukundan Sridharan, Hitay Ozbay, Raj Jain, " Guidelines for optimizing Multi-Level ECN, using fluid flow based TCP model" (http://www.cse.wustl.edu/~jain/papers.html)
- Sally Floyd, Ratul Mahajan, David Wetherall: RED-PD: RED with Preferential Dropping (http://www.cs.washington.edu/homes/ratul/red-pd/)
- A Generic Simple RED Simulator for educational purposes by Mehmet Suzen (http://code.google.com/p/guduz/)
- Approaches to Congestion Control in Packet Networks (http://utopia.duth.gr/~emamatas/jie2007.pdf)
- Papers in Congestion Control (http://www.ecse.rpi.edu/Homepages/shivkuma/research/cong-papers.html)
- Random Early Detection Homepage (http://www.icir.org/floyd/red.html)
- Explicit Congestion Notification Homepage (http://www.icir.org/floyd/ecn.html)
- TFRC Homepage (http://www.icir.org/tfrc/)
- AIMD-FC Homepage (http://www.ccs.neu.edu/home/ladrian/abstract/aimdfc.html)
- TCP congestion control simulation: Fast recovery (http://www.visualland.net/tcp_histrory.php?simu=tcp_fast_recovery&protocol=TCP&title=5.Fast recovery&ctype=1)

Path MTU Discovery

Path MTU Discovery (**PMTUD**) is a standardized technique in computer networking for determining the maximum transmission unit (MTU) size on the network path between two Internet Protocol (IP) hosts, usually with the goal of avoiding IP fragmentation. PMTUD is performed by routers in Internet Protocol Version 4 (IPv4),[1] while in IPv6 this function has been delegated to the end points of a communications session.[2]

Path MTU Discovery works by setting the *Don't Fragment* (DF) option bit in the IP headers of outgoing packets. Then, any device along the path whose MTU is smaller than the packet will drop it, and send back an Internet Control Message Protocol (ICMP) *Fragmentation Needed* (Type 3, Code 4) message containing its MTU, allowing the source host to reduce its Path MTU appropriately. The process repeats until the MTU is small enough to traverse the entire path without fragmentation.

If the Path MTU changes after the connection is set up and is lower than the previously determined Path MTU, the first large packet will cause an ICMP error and the new, lower Path MTU will be found. Conversely, if PMTUD finds that the path allows a larger MTU than is possible on the lower link, the OS will periodically reprobe to see if the path has changed and now allows larger packets. On Linux this timer is set by default to ten minutes,[3] and can be changed in /proc/sys/net/ipv4/route/mtu_expires.

Problems with PMTUD

Many network security devices block all ICMP messages for perceived security benefits,[4] including the errors that are necessary for the proper operation of PMTUD. This can result in connections that complete the TCP three-way handshake correctly, but then hang when data is transferred. This state is referred to as a *black hole connection*.[5]

Some implementations of PMTUD attempt to prevent this problem by inferring that large payload packets have been dropped due to MTU rather than because of link congestion. However, in order for the Transmission Control Protocol (TCP) to operate most efficiently, ICMP *Unreachable* messages (type 3) should be permitted. A robust method for PMTUD that relies on TCP or another protocol to probe the path with progressively larger packets has been standardized in RFC 4821.[6]

A workaround used by some routers is to change the maximum segment size (MSS) of all TCP connections passing through links with MTU lower than the Ethernet default of 1500. This is known as *MSS clamping*.[7]

References

[1] RFC 1191, *Path MTU Discovery*, J. Mogul, S. Deering (November 1990)
[2] RFC 1981, *Path MTU Discovery for IP version 6*, J. McCann, S. Deering, J. Mogul (August 1996)
[3] linux source code (http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.34.y.git;a=blob;f=net/ipv4/route.c;h=cb562fdd9b9a5342cee41f2cc642a8b4af9e30ce;hb=HEAD#l129) see line with "mtu_expires" 10 * 60 seconds
[4] "Prevent hacker probing by blocking ICMP traffic" (http://www.zdnet.com.au/prevent-hacker-probing-by-blocking-icmp-traffic-120280325.htm). .
[5] RFC 2923, *TCP Problems with Path MTU Discovery*, K. Lahey (September 2000)
[6] RFC 4821, *Packetization Layer Path MTU Discovery*, M. Mathis, J. Heffner (March 2007)
[7] "Circumventing Path MTU Discovery issues with MSS Clamping" (http://lartc.org/howto/lartc.cookbook.mtu-mss.html). .

External links

- Internet Engineering Task Force Path MTU Discovery (PMTUD) Working Group (http://www.ietf.org/html.charters/pmtud-charter.html)

Port number

In computer networking, a **port** is an application-specific or process-specific software construct serving as a communications endpoint in a computer's host operating system. A port is associated with an IP address of the host, as well as the type of protocol used for communication. The protocols that primarily use the ports are the Transport Layer protocols, such as the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) of the Internet Protocol Suite. A port is identified for each address and protocol by a 16-bit number, commonly known as the **port number**. The port number completes the destination address for a communications session. Thus, different IP addresses or protocols may use the same port number for communication, e.g. on a given host or interface UDP and TCP may use the same port number, or on a host with two interfaces, both addresses are associated with a port having the same number.

A range of well-known ports is reserved by convention to identify specific service types on a host. In the client-server model of application architecture ports are used to provide a multiplexing service on each port number that network clients connect to for service initiation, after which communication is reestablished on other connection-specific port number.

Technical details

Transport Layer protocols, such as the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), specify a source and destination port number in their packet headers. A port number is a 16-bit unsigned integer, thus ranging from 0 to 65535. A process associates its input or output channels via Internet sockets, a type of file descriptors, with a transport protocol, a port number and an IP address. This process is known as *binding*, and enables sending and receiving data via the network. The operating system's networking software has the task of transmitting outgoing data from all application ports onto the network, and forwarding arriving network packets to a process by matching the packet's IP address and port number. Only one process may bind to a specific IP address and port combination using the same transport protocol. Common application failures, sometimes called *port conflicts*, occur when multiple programs attempt to bind to the same port numbers on the same IP address using the same protocol.

Applications implementing common services often use specifically reserved, well-known port numbers for receiving service requests from client hosts. This process is known as *listening* and involves the receipt of a request on the well-known port and reestablishing one-to-one server-client communications on another *private* port, so that other clients may also contact the well-known service port. The well-known ports are defined by convention overseen by the Internet Assigned Numbers Authority (IANA).

The core network services, such as the World-Wide Web, typically use small port numbers less than 1024. In many operating systems special privileges are required for applications to bind to these ports, because these are often deemed critical to the operation of IP networks. Conversely, the client end of a connection typically uses a high port number allocated for short term use, therefore called an ephemeral port.

The port numbers are encoded in the transport protocol packet header, and they can be readily interpreted not only by the sending and receiving computers, but also by other components of the networking infrastructure. In particular, firewalls are commonly configured to differentiate between packets based on their source or destination port numbers. Port forwarding is an example application of this.

The practice of attempting to connect to a range of ports in sequence on a single computer is commonly known as port scanning. This is usually associated either with malicious cracking attempts or with network administrators looking for possible vulnerabilities to help prevent such attacks.

Port connection attempts are frequently monitored and logged by computers. The technique of port knocking uses a series of port connections (knocks) from a client computer to enable a server connection.

Examples

An example for the use of ports is the Internet mail system. A server used for sending and receiving email generally needs two services. The first service is used to transport email to and from other servers. This is accomplished with the Simple Mail Transfer Protocol (SMTP). The SMTP service application usually listens on TCP port 25 for incoming requests. The second service is the Post Office Protocol (POP) which is used by e-mail client applications on user's personal computers to fetch email messages from the server. The POP service listens on TCP port number 110. Both services may be running on the same host computer, in which case the port number distinguishes the service that was requested by a remote computer, be it a user's computer or another mail server.

While the listening port number of a server is well defined (IANA calls these the well known ports), the client's port number is often chosen from the dynamic port range (see below). In some applications, the client and the server each use specific port numbers assigned by the IANA. A good example of this is DHCP in which the client always uses UDP port 68 and the server always uses UDP port 67.

Use in URLs

Port numbers can occasionally be seen in the Uniform Resource Locator (URL) of a website or other services. By default, HTTP uses port 80 and HTTPS uses port 443, but a URL like `http://www.example.com:8000/path/` specifies that the web site is served by the HTTP server on port 8000. The active transport layer protocol ports may be discovered on many operating systems (Windows, Unix-like, z/OS) with the command line *netstat -a*.

Common port numbers

The Internet Assigned Numbers Authority (IANA) is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources. This includes the registration of commonly used port numbers for well-known Internet services.

The port numbers are divided into three ranges: the *well-known ports*, the *registered ports*, and the *dynamic* or *private ports*. The well-known ports are those from 0 through 1023. Examples include:

- *21*: File Transfer Protocol (FTP)
- *22*: Secure Shell (SSH)
- *23*: Telnet remote login service
- *25*: Simple Mail Transfer Protocol (SMTP)
- *53*: Domain Name System (DNS) service
- *80*: Hypertext Transfer Protocol (HTTP) used in the World Wide Web
- *110*: Post Office Protocol (POP)
- *119*: Network News Transfer Protocol (NNTP)
- *143*: Internet Message Access Protocol (IMAP)
- *161*: Simple Network Management Protocol (SNMP)
- *443*: HTTP Secure (HTTPS)

The registered ports are those from 1024 through 49151. IANA maintains the official list.[1] The dynamic or private ports are those from 49152 through 65535. One common use is for ephemeral ports.

History

The concept of port numbers was established by the early developers of the ARPANET in informal cooperation of software authors and system administrators.

The term *port number* was not yet used at this time. It was preceded by the use of the term *socket number* in the early development stages of the network. A socket number for a remote host was a 40-bit quantity.[2] The first 32 bits were similar to today's IPv4 address, but at the time the most-significant 8 bits were the host number. The least-significant portion of the socket number (bits 33 through 40) was an entity called *Another Eightbit Number*, abbreviated AEN,[3] today's port number.

On March 26, 1972, Vint Cerf and Jon Postel called for documenting the then current usages and establishing a socket number catalog in RFC 322. Network administrators were asked to submit a note or place a phone call, "*describing the function and socket numbers of network service programs at each HOST*".[4]

This catalog was subsequently published as RFC 433 in December 1972 and included a list of hosts and their port numbers and the corresponding function used at each host in the network. This first registry function served primarily as documentation of usage and indicated that port number usage was conflicting between some hosts for "*useful public services*".[3]

The document promised a resolution of the conflicts based on a standard that Postel had published in May 1972 in RFC 349, in which he first proposed official assignments of port numbers to network services and suggested a dedicated administrative function, which he call a *czar*, to maintain a registry.[5]

The 256 values of the AEN were divided into the following ranges:

- 0 through 63: network-wide standard functions
- 64 through 127: host-specific functions
- 128 through 239: reserved for future use
- 240 through 255: any experimental function

The Telnet service received the first official assignment of the value 1. In detail, the first set of assignments was:[5]

```
1  Telnet
3  File transfer
5  Remote job entry
7  Echo
9  Discard
```

In the early ARPANET, the AEN was also called a *socket name*,[6] and was used with the Initial Connection Protocol (ICP), a component of the Network Control Program (NCP)[7] [8] NCP was the forerunner of the modern Internet protocols. Today the terminology *service name* is still closely connected with port numbers, the former being text strings used in some network functions to represent a numerical port number.

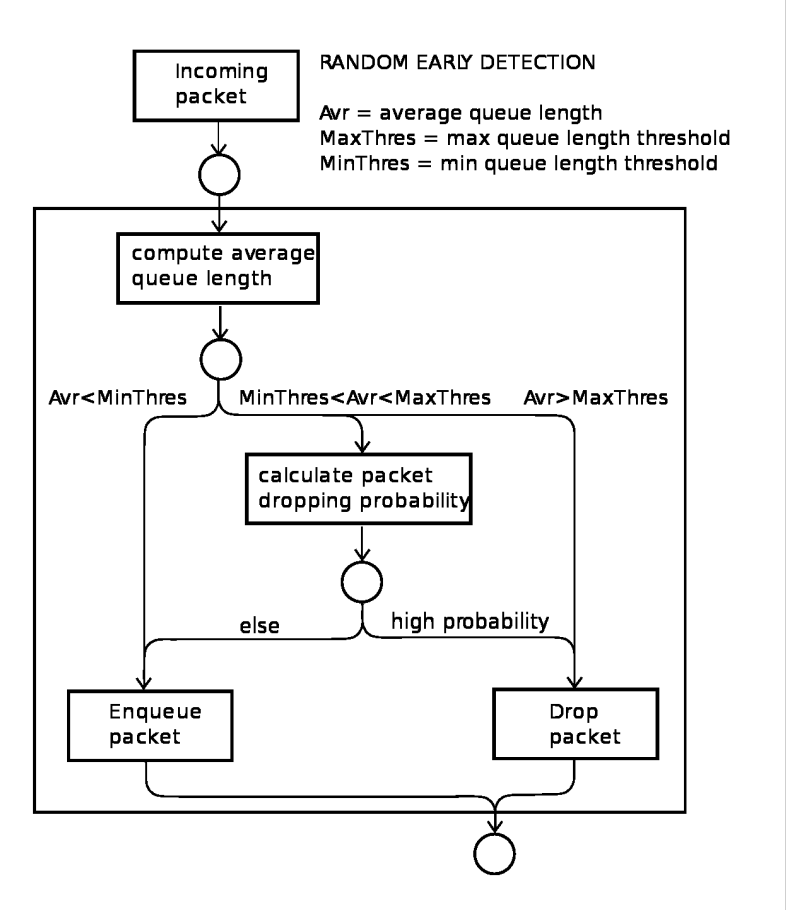
References

[1] "Port Numbers" (http://www.iana.org/assignments/port-numbers). The Internet Assigned Numbers Authority (IANA). .
[2] RFC 36, *Protocol Notes*, S. Crocker (16 March 1970)
[3] RFC 433, *Socket number list*, J. Postel, N. Neigus (22 December 1972)
[4] RFC 322, *Well Known Socket Numbers*, V. Cerf, J. Postel (26 March 1972)
[5] RFC 349, *Proposed Standard Socket Numbers* J. Postel (30 May 1972)
[6] RFC 197, *Initial Connection Protocol--Reviewed*, A. Shoshani, E. Harslem (14 July 1971)
[7] NIC 7104, *APRANET Protocol Handbook*
[8] Postel, Jon; Feinler, E. (1978). *ARPANET Protocol Handbook*. Menlo Park, CA: Network Information Center.

Random early detection

Random early detection (RED), also known as **random early discard** or **random early drop** is an active queue management algorithm. It is also a congestion avoidance algorithm.[1]

In the conventional tail drop algorithm, a router or other network component buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization as all TCP connections "hold back" simultaneously, and then step forward simultaneously. Networks become under-utilized and flooded by turns. RED addresses these issues.



Operation

RED monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped.

RED is more fair than tail drop, in the sense that it does not possess a bias against bursty traffic that uses only a small portion of the bandwidth. The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. Early detection helps avoid TCP global synchronization.

Problems with Classic RED

According to Van Jacobson, "there are not one, but two bugs in classic RED."[2] Improvements to the algorithm were developed, and a draft paper[3] was prepared, but the paper was never published, and the improvements were not widely disseminated or implemented.

Pure RED does not accommodate quality of service (QoS) differentiation. weighted RED (WRED) and RED with In and Out (RIO)[4] provide early detection with QoS considerations.

Other variants

In *Weighted RED* you can have different probabilities for different priorities (IP precedence, DSCP) and/or queues.[5]

The *Adaptive / Active RED* (ARED) algorithm[6] infers whether to make RED more or less aggressive based on the observation of the average queue length. If the average queue length oscillates around *min* threshold then early detection is too aggressive. On the other hand if the average queue length oscillates around *max* threshold then early detection is being too conservative. The algorithm changes the probability according to how aggressive it senses it has been discarding traffic.

See Srikant[7] for an in-depth account on these techniques and their analysis.

RRED: Robust RED

Robust Random Early Detection (RRED) algorithm was proposed to improve the TCP throughput against Denial-of-Service (DoS) attacks, particularly Low-rate Denial-of-Service [11] (LDoS) attacks. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under Low-rate Denial-of-Service (LDoS) attacks due to the oscillating TCP queue size caused by the attacks.[8] RRED algorithm can significantly improve the performance of TCP under Low-rate Denial-of-Service attacks.[8]

Notes and references

[1] Floyd, Sally; Jacobson, Van (August 1993). "Random Early Detection (RED) gateways for Congestion Avoidance" (http://www.icir.org/floyd/papers/red/red.html). *IEEE/ACM Transactions on Networking* **1** (4): 397–413. doi:10.1109/90.251892. . Retrieved 2008-03-16.

[2] Gettys, Jim (17 December 2010). "RED in a Different Light" (http://gettys.wordpress.com/2010/12/17/red-in-a-different-light/). *jg's Ramblings*. . Retrieved 27 December 2010.

[3] Jacobson, Van; Nichols, Kathy; Poduri, Kedar (30 September 1999). "RED in a Different Light" (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.9406). .

[4] D. Clark / J. Wroclawski (July, 1997). *An Approach to Service Allocation in the Internet* (http://tools.ietf.org/html/draft-clark-diff-svc-alloc-00). p. 12. . Retrieved 2011-05-27.

[5] Chao, H. Jonathan (2002). *Quality of service control in high speed networks*. 605 Third Avenue, New York, NY: John Wiley & Sons Inc.. doi:10.1002/0471224391.fmatter_indsub. ISBN 0-471-00397-2.

[6] Floyd, Sally; Gummadi, Ramakrishna; Shenker, Scott (2001-08-01). *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management* (http://citeseer.ist.psu.edu/floyd01adaptive.html). . Retrieved 2008-03-16.

[7] Srikant, Rayadurgam (2004). *The Mathematics of Internet Congestion Control* (http://www.springer.com/birkhauser/mathematics/book/978-0-8176-3227-4). Boston, MA, USA: Birkhäuser. ISBN 978-0-8176-3227-4. .

[8] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen, RRED: Robust RED Algorithm to Counter Low-rate Denial-of-Service Attacks (http://sites.google.com/site/cwzhangres/home/files/RREDRobustREDAlgorithmtoCounterLow-rateDenial-of-ServiceAttacks.pdf?attredirects=0), IEEE Communications Letters, vol. 14, pp. 489-491, 2010. Ref (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5456075)

External links

- RED (Random Early Detection) Queue Management (http://www.icir.org/floyd/red.html). Author: Sally Floyd
- Guduz (http://code.google.com/p/guduz/) – A Simple Random Early Detection (RED) Simulator
- Variants of RED (http://www.ee.ust.hk/~heixj/publication/thesis/node37.html). Author: Hei Xiao Jun
- Recent Publications in Random Early Detection (RED) schemes (http://sites.google.com/site/cwzhangres/home/posts/recentpublicationsinrandomearlydetectionredschemes)

Retransmission (data networks)

Retransmission, essentially identical with Automatic repeat request (ARQ), is the resending of packets which have been either damaged or lost. It is a term that refers to one of the basic mechanisms used by protocols operating over a packet switched computer network to provide reliable communication (such as that provided by a reliable byte stream, for example TCP).

Such networks are usually 'unreliable', meaning they offer no guarantees that they will not delay, damage, or lose packets, or deliver them out of order.

Protocols which provide reliable communication over such networks use a combination of acknowledgments (i.e. an explicit receipt from the destination of the data), retransmission of missing and/or damaged packets (usually initiated by a time-out), and checksums to provide that reliability.

There are several forms of acknowledgement which can be used alone or together in networking protocols:

- Positive Acknowledgement: the receiver explicitly notifies the sender which packets, messages, or segments were received correctly which may implicitly inform the sender which packets were not received even though they were sent and thus may need to be retransmitted. Positive Acknowledgment with Re-Transmission (PAR), is a method used by TCP (RFC 793) to verify receipt of transmitted data. PAR operates by re-transmitting data at an established period of time until the receiving host acknowledges reception of the data.
- Negative Acknowledgment (NACK): the receiver explicitly notifies the sender which packets, messages, or segments were received incorrectly and thus may need to be retransmitted (RFC 4077).
- Selective Acknowledgment (SACK): the receiver explicitly lists which packets, messages, or segments in a stream are acknowledged (either negatively or positively). Positive selective acknowledgment is an option in TCP (RFC 2018) that is useful in Satellite Internet access (RFC 2488).
- Cumulative Acknowledgment: the receiver acknowledges that it correctly received a packet, message, or segment in a stream which implicitly informs the sender that the previous packets were received correctly. TCP uses cumulative acknowledgment with its TCP sliding window.

Retransmission is a very simple concept. Whenever one party sends something to the other party, it retains a copy of the data it sent until the recipient has acknowledged that it received it. In a variety of circumstances, e.g.:

- if no such acknowledgment is forthcoming within a reasonable time, the time-out, or
- the sender discovers, often through some out of band means, that the transmission was unsuccessful, or
- if the receiver knows that expected data has not arrived, and so notifies the sender, or
- if the receiver knows that the data has arrived, but in a damaged condition, and indicates that to the sender, the sender simply automatically retransmits the data (which it still has a copy of).

Round-trip delay time

In telecommunications, the **round-trip delay time (RTD)** or **round-trip time (RTT)** is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the transmission times between the two points of a signal.

- In the context of computer networks, the signal is generally a data packet, and the RTT is also known as the **ping time**. An internet user can determine the RTT by using the ping command.
- In space technology, the *round-trip delay time* or *round trip light time* is the time light (and hence any signals) takes to go to the spacecraft and to return.

Network links with both a high bandwidth and a high RTT can have a very large amount of data (the bandwidth-delay product) "in flight" at any given time. Such "long fat pipes" require a special protocol design. One example is the TCP window scale option.

The RTT was originally estimated in TCP by:

$$\mathbf{RTT = (\alpha \cdot Old_RTT) + ((1 - \alpha) \cdot New_Round_Trip_Sample)}$$ [1]

Where α is constant weighting factor($0 \leq \alpha < 1$). Choosing a value α close to 1 makes the weighted average immune to changes that last a short time (e.g., a single segment that encounters long delay). Choosing a value for α close to 0 makes the weighted average respond to changes in delay very quickly.

This was improved by the Jacobson/Karels algorithm, which takes standard deviation into account as well.

Once a new RTT is calculated, it is entered into the equation above to obtain an average RTT for that connection, and the procedure continues for every new calculation.

References

[1] Comer, Douglas. Internetworking with TCP/IP. Page 226. Upper Saddle River, N.J.: Prentice Hall, 2000. Print.

Silly window syndrome

Silly window syndrome is a problem in computer networking caused by poorly-implemented TCP flow control. If a server with this problem is unable to process all incoming data, it requests that its clients reduce the amount of data they send at a time (the "window" setting on a TCP packet). If the server continues to be unable to process all incoming data, the window becomes smaller and smaller, sometimes to the point that the data transmitted is smaller than the packet header, making data transmission extremely inefficient. The name of this problem is due to the window size shrinking to a "silly" value.

Since there is a certain amount of overhead associated with processing each packet, the increased number of packets means increased overhead to process a decreasing amount of data. The end result is thrashing.

Solution

When there is no synchronization between the sender and receiver regarding capacity of the flow of data or the size of the packet, the window syndrome problem is created. When the silly window syndrome is created by the sender, Nagle's algorithm is used. Nagle's solution requires that the sender sends the first segment even if it is a small one, then the sender waits until an ACK is received or a maximum sized segment (MSS) is accumulated.

When the silly window syndrome is created by the receiver, David D Clark's solution is used. Clark's solution closes the window until another segment of maximum segment size (MSS) can be received or the buffer is half empty.

Send-side silly window avoidance

A heuristic method where the send TCP must allow the sending application to make "write" calls, and collect the data transferred in each call before transmitting it into a large segment. The sending TCP delays sending segments until it can accumulate reasonable amounts of data, which is known as *clumping*.

Receive-side silly window avoidance

A heuristic method that a receiver uses to maintain an internal record of the available window, and delay advertising an increase in window size to the sender until it can advance a significant amount. This amount depends on the receiver's buffer size and maximum segment size. By using this method, it prevents small window advertisements where received applications extract data octets slowly.

References

- Comer, Douglas E. (2006). *Internetworking with TCP/IP* (5 ed.). Prentice Hall: Upper Saddle River, NJ.

External links

- Explanation of the silly window syndrome [1]

References

[1] http://www.tcpipguide.com/free/t_TCPSillyWindowSyndromeandChangesTotheSlidingWindow.htm

Sliding window protocol

A **sliding window protocol** is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol (TCP).

Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit of the size of the sequence numbers that can be required.

By placing limits on the number of packets that can be transmitted or received at any given time, a sliding window protocol allows an unlimited number of packets to be communicated using fixed-size sequence numbers.

For the highest possible throughput, it is important that the transmitter is not forced to stop sending by the sliding window protocol earlier than one round-trip delay time (RTT). The limit on the amount of data that it can send before stopping to wait for an acknowledgment should be larger than the bandwidth-delay product of the communications link. If it is not, the protocol will limit the effective bandwidth of the link.

Motivation

In any communication protocol based on automatic repeat request for error control, the receiver must acknowledge received packets. If the transmitter does not receive an acknowledgment within a reasonable time, it re-sends the data.

A transmitter that does not hear an acknowledgment cannot know if the receiver actually received the packet; it may be that the packet was lost in transmission (or damaged; if error detection finds an error, the packet is ignored), or it may be that an acknowledgment was sent, but it was lost. In the latter case, the receiver must acknowledge the retransmission, but must otherwise ignore it.

Likewise, the receiver is usually uncertain about whether its acknowledgments are being received.

Protocol operation

The transmitter and receiver each have a current sequence number n_t and n_r, respectively. They each also have a window size w_t and w_r. The window sizes may vary, but in simpler implementations they are fixed. The window size must be greater than zero for any progress to be made.

As typically implemented, n_t is the next packet to be transmitted, i.e. the sequence number of the first packet not yet transmitted. Likewise, n_r is the first packet not yet received. Both numbers are monotonically increasing with time; they only ever increase.

The receiver may also keep track of the highest sequence number not yet received; the variable n_s is one more than the sequence number of the highest sequence number received. For simple receivers that only accept packets in order (w_r=1), this is the same as n_r, but can be greater if w_r>1. Note the distinction: all packets below n_r have been received, no packets above n_s have been received, and between n_r and n_s, some packets have been received.

When the receiver receives a packet, it updates its variables appropriately and transmits an acknowledgment with the new n_r. The transmitter keeps track of the highest acknowledgment it has received n_a. The transmitter knows that all packets up to, but not including n_a have been received, but is uncertain about packets between n_a and n_s; i.e. $n_a \le n_r \le n_s$.

The sequence numbers always obey the rule that $n_a \le n_r \le n_s \le n_t \le n_a + w_t$. That is:

- $n_a \le n_r$: The highest acknowledgement received by the transmitter cannot be higher than the highest n_r acknowledged by the receiver.

- $n_r \le n_s$: The span of fully received packets cannot extend beyond the end of the partially received packets.
- $n_s \le n_t$: The highest packet received cannot be higher than the highest packet sent.
- $n_t \le n_a + w_t$: The highest packet sent is limited by the highest acknowledgement received and the transmit window size.

Transmitter operation

Whenever the transmitter has data to send, it may transmit up to w_t packets ahead of the latest acknowledgment n_a. That is, it may transmit packet number n_t as long as $n_t < n_a + w_t$.

In the absence of a communication error, the transmitter soon receives an acknowledgment for all the packets it has sent, leaving n_a equal to n_t. If this does not happen after a reasonable delay, the transmitter must retransmit the packets between n_a and n_t.

Techniques for defining "reasonable delay" can be extremely elaborate, but they only affect efficiency; the basic reliability of the sliding window protocol does not depend on the details.

Receiver operation

Every time a packet numbered x is received, the receiver checks to see if it falls in the receive window, $n_r \le x < n_s + w_r$. (The simplest receivers only have to keep track of one value $n_r = n_s$.) If it falls within the window, the receiver accepts it. If it is numbered n_r, the receive sequence number is increased by 1, and possibly more if further consecutive packets were previously received and stored. If $x > n_r$, the packet is stored until all preceding packets have been received.[1] If $x \ge n_s$, the latter is updated to $n_s = x+1$.

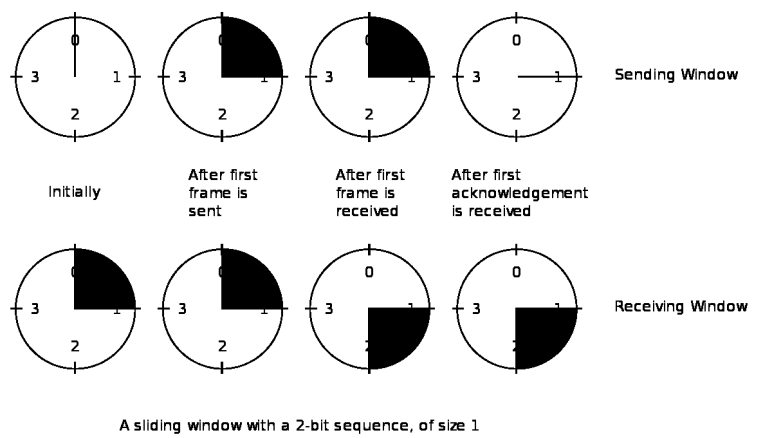
If the packet's number is not within the receive window, the receiver discards it and does not modify n_r or n_s.

Whether the packet was accepted or not, the receiver transmits an acknowledgment containing the current n_r. (The acknowledgment may also include information about additional packets received between n_r or n_s, but that only helps efficiency.)

Note that there is no point having the receive window w_r larger than the transmit window w_t, because there is no need to worry about receiving a packet that will never be transmitted; the useful range is $1 \le w_r \le w_t$.

Sequence number range required

So far, the protocol has been described as if sequence numbers are of unlimited size, ever-increasing. However, rather than transmitting the full sequence number x in messages, it is possible to transmit only $x \bmod N$, for some finite N. (N is usually a power of 2.)



Sequence numbers modulo 4, with w_r=1. Initially, n_t=n_r=0

For example, the transmitter will only receive acknowledgments in the range n_a to n_t, inclusive. Since it guarantees that $n_t - n_a \le w_t$, there are at most w_t+1 possible sequence numbers that could arrive at any given time. Thus, the transmitter can unambiguously decode the sequence number as long as $N > w_t$.

A stronger constraint is imposed by the receiver. The operation of the protocol depends on the receiver being able to reliably distinguish new packets (which should be accepted and processed) from retransmissions of old packets (which should be discarded, and the last acknowledgment retransmitted). This can be done given knowledge of the transmitter's window size. After receiving a packet numbered x, the receiver knows that $x < n_a+w_t$, so $n_a > x-w_t$. Thus, packets numbered $x-w_t$ will never again be retransmitted.

The lowest sequence number we will ever receive in future is n_s-w_t

The receiver also knows that the transmitter's n_a cannot be higher than the highest acknowledgment ever sent, which is n_r. So the highest sequence number we could possibly see is $n_r+w_t \leq n_s+w_t$.

Thus, there are $2w_t$ different sequence numbers that the receiver can receive at any one time. It might therefore seem that we must have $N \geq 2w_t$. However, the actual limit is lower.

The additional insight is that the receiver does not need to distinguish between sequence numbers that are too low (less than n_r) or that are too high (greater than or equal to n_s+w_r). In either case, the receiver ignores the packet except to retransmit an acknowledgment. Thus, it is only necessary that $N \geq w_t+w_r$. As it is common to have $w_r<w_t$ (e.g. see Go-Back-N below), this can permit larger w_t within a fixed N.

Examples

The simplest sliding window: stop-and-wait

Although commonly distinguished from the sliding-window protocol, the stop-and-wait ARQ protocol is actually the simplest possible implementation of it. The transmit window is 1 packet, and the receive window is 1 packet. Thus, N=1+1=2 possible sequence numbers (conveniently represented by a single bit) are required.

Ambiguity example

The transmitter alternately sends packets marked "odd" and "even". The acknowledgments likewise say "odd" and "even". Suppose that the transmitter, having sent an odd packet, did not wait for an odd acknowledgment, and instead immediately sent the following even packet. It might then receive an acknowledgment saying "expecting an odd packet next". This would leave the transmitter in a quandary: has the receiver received both of the packets, or neither?

Go-Back-N

Go-Back-N ARQ is the sliding window protocol with w_t>1, but a fixed w_r=1. The receiver refuses to accept any packet but the next one in sequence. If a packet is lost in transit, following packets are ignored until the missing packet is retransmitted, a minimum loss of one round trip time. For this reason, it is inefficient on links that suffer frequent packet loss.

Ambiguity example

Suppose that we are using a 3-bit sequence number, such as is typical for HDLC. This gives $N=2^3=8$. Since w_r=1, we must limit $w_t \leq 7$. This is because, after transmitting 7 packets, there are 8 possible results: Anywhere from 0 to 7 packets could have been received successfully. This is 8 possibilities, and the transmitter needs enough information in the acknowledgment to distinguish them all.

If the transmitter sent 8 packets without waiting for acknowledgment, it could find itself in a quandary similar to the stop-and-wait case: does the acknowledgment mean that all 8 packets were received successfully, or none of them?

Selective repeat

The most general case of the sliding window protocol is Selective Repeat ARQ. This requires a much more capable receiver, which can accept packets with sequence numbers higher than the current n_r and store them until the gap is filled in.

The advantage, however, is that it is not necessary to discard following correct data for one round-trip time before the transmitter can be informed that a retransmission is required. This is therefore preferred for links with low reliability and/or a high bandwidth-delay product.

The window size w_r need only be larger than the number of *consecutive* lost packets that can be tolerated. Thus, small values are popular; w_r=2 is common.

Ambiguity example

The extremely popular HDLC protocol uses a 3-bit sequence number, and has optional provision for selective repeat. However, if selective repeat is to be used, the requirement that $n_t+n_r \leq 8$ must be maintained; if w_r is increased to 2, w_t must be decreased to 6.

Suppose that w_r =2, but an unmodified transmitter is used with w_t =7, as is typically used with the go-back-N variant of HDLC. Further suppose that the receiver begins with n_r =n_s =0.

Now suppose that the receiver sees the following series of packets (all modulo 8):

0 1 2 3 4 5 6 *(pause)* 0

Because w_r =2, the receiver will accept and store the final packet 0 (thinking it is packet 8 in the series), while requesting a retransmission of packet 7. However, it is also possible that the transmitter failed to receive any acknowledgments and has retransmitted packet 0. In this latter case, the receiver would accept the wrong packet as packet 8. . The solution is for the transmitter to limit $w_t \leq 6$. With this restriction, the receiver knows, after receiving packet 6, that the transmitter's $n_a \geq 1$, and thus the following packet numbered 0 must be packet 8. If all acknowledgements were lost, then the transmitter would have to stop after packet 5.

Extensions

There are many ways that the protocol can be extended:

- The above examples assumed that packets are never reordered in transmission; they may be lost in transit (error detection makes corruption equivalent to loss), but will never appear out of order.

The protocol can be extended to support packet reordering, as long as the distance can be bounded; the sequence number modulus *N* must be expanded by the maximum misordering distance.

- It is possible to not acknowledge every packet, as long as an acknowledgment is sent after not receiving any packets for a while. For example, TCP normally acknowledges every second packet.
- It is common to inform the transmitter immediately if a gap in the packet sequence is detected. HDLC has a special REJ (reject) packet for this.
- The transmit and receive window sizes may be changed during communication, as long as their sum remains within the limit of *N*. In particular, it is common to reduce the transmit window size to slow down transmission to match the link's speed, avoiding saturation or congestion.
- One common simplification of selective-repeat is so called SREJ-REJ ARQ. This operates with w_r=2 and accepts following packets, but only allows a single lost packet; if a second packet is lost, no more packets are buffered. (I.e. w_r=1 while waiting.) This gives most of the performance benefits of the full selective-repeat protocol, with a simpler implementation.

References

[1] Peterson, Larry L. & Davie, Bruce S. " Computer Networks: A Systems Approach (http://www.ebook3000.com/Computer-Networks--A-Systems-Approach_8497.html)", Morgan Kaufmann, 2000. ISBN 1-55860-577-0

- Comer, Douglas E. "Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture", Prentice Hall, 1995. ISBN 0132169878
- Peterson, Larry L. & Davie, Bruce S. "Computer Networks: A Systems Approach", Morgan Kaufmann, 2000. ISBN 15586051428888

External links

- RFC 1323 - TCP Extensions for High Performance
- TCP window scaling and broken routers (http://lwn.net/Articles/92727/), 2004
- Sliding Window Demo (http://www2.rad.com/networks/2004/sliding_window/) (Flash required)
- Sliding Window Demo (http://www.humboldt.edu/~aeb3/telecom/SlidingWindow.html) (Shockwave required)
- TCP Sliding Window simulation (interactive animation) (http://www.visualland.net/tcp_histrory.php?simu=tcp_swnd&protocol=TCP&title=2.Sliding Window&ctype=1)

Slow-start

Slow-start is part of the congestion control strategy used by TCP, the data transmission protocol used by many Internet applications. Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion.

Algorithm

Slow-start is one of the algorithms that TCP uses to control congestion inside the network. It is also known as the exponential growth phase.

During the exponential growth phase, slow-start works by increasing the TCP congestion window each time the acknowledgment is received. It increases the window size by the number of segments acknowledged. This happens until either an acknowledgment is not received for some segment or a predetermined threshold value is reached. If a loss event occurs, TCP assumes that it is due to network congestion and takes steps to reduce the offered load on the network. Once a loss event has occurred or the threshold has been reached, TCP enters the linear growth (congestion avoidance) phase. At this point, the window is increased by 1 segment for each RTT. This happens until a loss event occurs.

Although the strategy is referred to as "slow-start", its congestion window growth is quite aggressive, more aggressive than the congestion avoidance phase (Jacobson, 1988). Before "slow start" was introduced in TCP, the initial pre-congestion avoidance phase was even faster.

Basic slow-start

The algorithm begins in the exponential growth phase initially with a congestion window size (cwnd) of 1 or 2 segments and increases it by 1 Segment Size (SS) **for each ACK** received. Since the receiver typically sends an ACK for every two segments, this behavior effectively doubles the window size each round trip of the network. This behavior continues until the congestion window size (cwnd) reaches the size of the receiver's advertised window or until a loss occurs.

When a loss occurs half of the current cwnd is saved as a Slow Start Threshold (SSThresh) and slow start begins again from its initial cwnd. Once the cwnd reaches the SSThresh TCP goes into congestion avoidance mode where each ACK increases the cwnd by SS*SS/cwnd. This results in a linear increase of the cwnd.

See RFC 2001.[1]

Fast recovery

There is a variation to the slow-start algorithm known as fast recovery, which uses fast retransmit followed by congestion avoidance. In the fast recovery algorithm, during congestion avoidance mode, when packets (detected through 3 duplicate ACKs) are not received, the congestion window size is reduced to the slow-start threshold, rather than the smaller initial value.

Problems

Slow-start assumes that unacknowledged segments are due to network congestion. While this is an acceptable assumption for many networks, segments may be lost for other reasons, such as poor data link layer transmission quality. Thus, slow-start can perform poorly in situations with poor reception, such as wireless networks.

The slow-start protocol performs badly for short-lived connections. Older web browsers would create many consecutive short-lived connections to the web server, and would open and close the connection for each file requested. This kept most connections in the slow start mode, which resulted in poor response time. To avoid this problem, modern browsers either open multiple connections simultaneously or reuse one connection for all files requested from a particular web server.

References

[1] W. Richard Stevens (January 1997). "RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms" (http://tools.ietf.org/html/rfc2001). Network Working Group of the IETF. . Please note that RFC 2001 has since been obsoleted by "RFC 2581 – TCP Congestion Control". RFC 2581 was then updated by "RFC 3390 – Increasing TCP's Initial Window" and then obsoleted by "RFC 5681 – TCP Congestion Control".

External links

- TCP Slow Start Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/TCP_Slow_Start.pdf)

TCP congestion avoidance algorithm

Transmission Control Protocol (TCP) uses a network congestion avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, with other schemes such as slow-start in order to achieve congestion avoidance.

The **TCP congestion avoidance algorithm** is the primary basis for congestion control in the Internet.[1] [2] [3] [4] [5]

Naming history

Two such variations are those offered by TCP Tahoe and Reno. The two algorithms were retrospectively named after the 4.3BSD operating system in which each first appeared (which were themselves named after Lake Tahoe and the city of Reno, Nevada). The “Tahoe” algorithm first appeared in 4.3BSD-Tahoe (which was made to support the CCI Power 6/32 “Tahoe” minicomputer), and was made available to non-AT&T licensees as part of the “4.3BSD Networking Release 1”; this ensured its wide distribution and implementation. Improvements, described below, were made in 4.3BSD-Reno and subsequently released to the public as “Networking Release 2” and later 4.4BSD-Lite. The “TCP Foo” names for the algorithms appear to have originated in a 1996 paper by Kevin Fall and Sally Floyd.[6]

TCP Tahoe and Reno

To avoid congestion collapse, TCP uses a multi-faceted congestion control strategy. For each connection, TCP maintains a *congestion window*, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called *slow start*[7] to increase the congestion window after a connection is initialized and after a timeout. It starts with a window of two times the maximum segment size (MSS). Although the initial rate is low, the rate of increase is very rapid: **for every packet** acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round trip time (RTT). When the congestion window exceeds a threshold *ssthresh* the algorithm enters a new state, called congestion avoidance. In some implementations (*e.g.*, Linux), the initial ssthresh is large, and so the first slow start usually ends after a loss. However, ssthresh is updated at the end of each slow start, and will often affect subsequent slow starts triggered by timeouts.

Congestion avoidance: As long as non-duplicate ACKs are received, the congestion window is additively increased by one MSS every round trip time. When a packet is lost, the likelihood of duplicate ACKs being received is very high (it's possible though unlikely that the stream just underwent extreme packet reordering, which would also prompt duplicate ACKs). The behavior of Tahoe and Reno differ in how they detect and react to packet loss:

- Tahoe: Triple duplicate ACKS are treated the same as a timeout. Tahoe will perform "fast retransmit", reduce congestion window to 1 MSS, and reset to slow-start state.[8]
- Reno: If three duplicate ACKs are received (i.e., four ACKs acknowledging the same packet, which are not piggybacked on data, and do not change the receiver's advertised window), Reno will halve the congestion window, perform a fast retransmit, and enter a phase called Fast Recovery. If an ACK times out, slow start is used as it is with Tahoe.[8]

Fast Recovery. (Reno Only) In this state, TCP retransmits the missing packet that was signaled by three duplicate ACKs, and waits for an acknowledgment of the entire transmit window before returning to congestion avoidance. If there is no acknowledgment, TCP Reno experiences a timeout and enters the slow-start state.

Both algorithms reduce congestion window to 1 MSS on a timeout event.

TCP Vegas

Until the mid 1990s, all of TCP's set timeouts and measured round-trip delays were based upon only the last transmitted packet in the transmit buffer. University of Arizona researchers Larry Peterson and Lawrence Brakmo introduced TCP Vegas, in which timeouts were set and round-trip delays were measured for every packet in the transmit buffer. In addition, TCP Vegas uses additive increases in the congestion window. This variant was not widely deployed outside Peterson's laboratory.

However, TCP Vegas was deployed as default congestion control method for DD-WRT firmwares v24 SP2.

TCP New Reno

TCP New Reno [9] improves retransmission during the fast recovery phase of TCP Reno. During fast recovery, for every duplicate ACK that is returned to TCP New Reno, a new unsent packet from the end of the congestion window is sent, to keep the transmit window full. For every ACK that makes partial progress in the sequence space, the sender assumes that the ACK points to a new hole, and the next packet beyond the ACKed sequence number is sent.

Because the timeout timer is reset whenever there is progress in the transmit buffer, this allows New Reno to fill large holes, or multiple holes, in the sequence space - much like TCP SACK. Because New Reno can send new packets at the end of the congestion window during fast recovery, high throughput is maintained during the hole-filling process, even when there are multiple holes, of multiple packets each. When TCP enters fast recovery it records the highest outstanding unacknowledged packet sequence number. When this sequence number is acknowledged, TCP returns to the congestion avoidance state.

A problem occurs with New Reno when there are no packet losses but instead, packets are reordered by more than 3 packet sequence numbers. When this happens, New Reno mistakenly enters fast recovery, but when the reordered packet is delivered, ACK sequence-number progress occurs and from there until the end of fast recovery, every bit of sequence-number progress produces a duplicate and needless retransmission that is immediately ACKed.

New Reno performs as well as SACK at low packet error rates, and substantially outperforms Reno at high error rates.

TCP Hybla

TCP Hybla [10] aims to eliminate penalization of TCP connections that incorporate a high-latency terrestrial or satellite radio link, due to their longer round trip times. It stems from an analytical evaluation of the congestion window dynamics, which suggests the necessary modifications to remove the performance dependence on RTT.

TCP BIC

Binary Increase Congestion control is an implementation of TCP with an optimized congestion control algorithm for high speed networks with high latency (called **LFN**, long fat networks, in RFC 1072 [11]). BIC is used by default in Linux kernels 2.6.8 through 2.6.18.

TCP CUBIC

CUBIC is a less aggressive and more systematic derivative of BIC, in which the window is a cubic function of time since the last congestion event, with the inflection point set to the window prior to the event. CUBIC is used by default in Linux kernels since version 2.6.19.

Compound TCP

Compound TCP is a Microsoft implementation of TCP which maintains two different congestion windows simultaneously, with the goal of achieving good performance on LFNs while not impairing fairness. It has been widely deployed with Microsoft Windows Vista and Windows Server 2008 and has been ported to older Microsoft Windows versions as well as Linux.

Other TCP congestion avoidance algorithms

- FAST TCP
- H-TCP
- Data Center TCP
- High Speed TCP
- HSTCP-LP [12]
- TCP-Illinois
- TCP-LP [12]
- TCP SACK
- Scalable TCP [13]
- TCP Veno
- Westwood
- Westwood+
- XCP [14]
- YeAH-TCP [15]
- TCP-FIT [16]
- Congestion Avoidance with Normalized Interval of Time (CANIT)[17]

TCP New Reno is the most commonly implemented algorithm, SACK support is very common and is an extension to Reno/New Reno. Most others are competing proposals which still need evaluation. Starting with 2.6.8 the Linux kernel switched the default implementation from Reno to BIC. The default implementation was again changed to CUBIC in the 2.6.19 version.

When the per-flow product of bandwidth and latency increases, regardless of the queuing scheme, TCP becomes inefficient and prone to instability. This becomes increasingly important as the Internet evolves to incorporate very high-bandwidth optical links.

TCP Interactive (iTCP) [18] allows applications to subscribe to TCP events and respond accordingly enabling various functional extensions to TCP from outside TCP layer. Most TCP congestion schemes work internally. iTCP additionally enables advanced applications to directly participate in congestion control such as to control the source generation rate.

Zeta-TCP detects the congestions from both the latency and loss rate measures, and applies different CWND backoff strategies based on the likelihood of the congestions to maximize the goodput. It also has a couple of other improvements to accurately detect the packet losses, avoiding RTO retransmission; and accelerate/control the inbound (download) traffic.

References

[1] Van Jacobson, Michael J. Karels. Congestion Avoidance and Control (http://citeseer.ist.psu.edu/484335.html) (1988). *Proceedings of the Sigcomm '88 Symposium*, vol.18(4): pp.314–329. Stanford, CA. August, 1988. This paper originated many of the congestion avoidance algorithms used in TCP/IP.

[2] RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

[3] RFC 2581 - TCP Congestion Control

[4] RFC 3390 - TCP Increasing TCP's Initial Window

[5] TCP Congestion Avoidance Explained via a Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/TCP_Congestion_Avoidance.pdf)

[6] Fall, Kevin; Sally Floyd (July 1996). "Simulation-based Comparisons of Tahoe, Reno and SACK TCP" (ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z) (PostScript). *Computer Communications Review*. .

[7] Jacobson, Van; Karels, Michael (1988). "Congestion Avoidance and Control" (http://ee.lbl.gov/papers/congavoid.pdf). *ACM SIGCOMM Computer Communication Review* **25** (1): 157–187. doi:10.1145/205447.205462. .

[8] Kurose & Ross 2008, p. 284.

[9] http://www.faqs.org/rfcs/rfc3782.html

[10] http://hybla.deis.unibo.it/

[11] http://tools.ietf.org/html/rfc1072

[12] http://www.ece.rice.edu/networks/TCP-LP/

[13] http://www.deneholme.net/tom/scalable/

[14] http://www.isi.edu/isi-xcp/

[15] http://wil.cs.caltech.edu/pfldnet2007/paper/YeAH_TCP.pdf

[16] http://media.cs.tsinghua.edu.cn/~multimedia/tcp-fit/

[17] *An analytical study of CANIT algorithm in TCP protocol*. doi:10.1145/605521.605530.

[18] http://www.medianet.kent.edu/itcp/main.html

Kurose, James; Ross, Keith (2008). *Computer Networking - A Top-Down Approach* (4th th ed.). Addison Wesley. ISBN 978-0136079675.

Afanasyev, A.; N. Tilley, P. Reiher, and L. Kleinrock (2010). "Host-to-host congestion control for TCP" (http://lasr.cs.ucla.edu/afanasyev/data/files/Afanasyev/Host-to-host congestion control for TCP.pdf). *IEEE Communication Surveys and Tutorials* **12** (3).

External links

- Congestion avoidance simulation (http://www.visualland.net/tcp_histrory.php?simu=tcp_swnd&protocol=TCP&title=2.Sliding Window&ctype=1)

Transmission Control Protocol

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. TCP is the protocol that major Internet applications such as the World Wide Web, email, remote administration and file transfer rely on. Other applications, which do not require reliable data stream service, may use the User Datagram Protocol (UDP), which provides a datagram service that emphasizes reduced latency over reliability.

Historical origin

In May 1974 the Institute of Electrical and Electronic Engineers (IEEE) published a paper entitled "*A Protocol for Packet Network Interconnection.*"[1] The paper's authors, Vint Cerf and Bob Kahn described an internetworking protocol for sharing resources using packet-switching among the nodes. A central control component of this model was the *Transmission Control Program* that incorporated both connection-oriented links and datagram services between hosts. The monolithic Transmission Control Program was later divided into a modular architecture consisting of the *Transmission Control Protocol* at the connection-oriented layer and the *Internet Protocol* at the internetworking (datagram) layer. The model became known informally as *TCP/IP*, although formally it was henceforth called the *Internet Protocol Suite*.

Network function

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details.

IP works by exchanging pieces of information called packets. A packet is a sequence of octets and consists of a *header* followed by a *body*. The header describes the packet's destination and, optionally, the routers to use for forwarding until it arrives at its destination. The body contains the data IP is transmitting.

Due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the application program. Thus, TCP abstracts the application's communication from the underlying networking details.

TCP is utilized extensively by many of the Internet's most popular applications, including the World Wide Web (WWW), E-mail, File Transfer Protocol, Secure Shell, peer-to-peer file sharing, and some streaming media applications.

TCP is optimized for accurate delivery rather than timely delivery, and therefore, TCP sometimes incurs relatively long delays (in the order of seconds) while waiting for out-of-order messages or retransmissions of lost messages. It is not particularly suitable for real-time applications such as Voice over IP. For such applications, protocols like the Real-time Transport Protocol (RTP) running over the User Datagram Protocol (UDP) are usually recommended instead.[2]

TCP is a reliable stream delivery service that guarantees delivery of a data stream sent from one host to another without duplication or losing data. Since packet transfer is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record

of each packet it sends, and waits for acknowledgment before sending the next packet. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires. The timer is needed in case a packet gets lost or corrupted.[2]

TCP consists of a set of rules: for the protocol, that are used with the Internet Protocol, and for the IP, to send data "in a form of message units" between computers over the Internet. While IP handles actual delivery of the data, TCP keeps track of the individual units of data transmission, called *segments*, that a message is divided into for efficient routing through the network. For example, when an HTML file is sent from a Web server, the TCP software layer of that server divides the sequence of octets of the file into segments and forwards them individually to the IP software layer (Internet Layer). The Internet Layer encapsulates each TCP segment into an IP packet by adding a header that includes (among other data) the destination IP address. Even though every packet has the same destination address, they can be routed on different paths through the network. When the client program on the destination computer receives them, the TCP layer (Transport Layer) reassembles the individual segments and ensures they are correctly ordered and error free as it streams them to an application.

TCP segment structure

Transmission Control Protocol accepts data from a data stream, segments it into chunks, and adds a TCP header creating a TCP segment. The TCP segment is then encapsulated into an Internet Protocol (IP) datagram. A TCP segment is "the packet of information that TCP uses to exchange data with its peers." [3]

The term *TCP packet*, though sometimes informally used, is not in line with current terminology, where *segment* refers to the TCP PDU (Protocol Data Unit), *datagram*[4] to the IP PDU and *frame* to the data link layer PDU:

> Processes transmit data by calling on the TCP and passing buffers of data as arguments. The TCP packages the data from these buffers into segments and calls on the internet module [e.g. IP] to transmit each segment to the destination TCP.[5]

A TCP segment consists of a segment *header* and a *data* section. The TCP header contains 10 mandatory fields, and an optional extension field (*Options*, pink background in table).

The data section follows the header. Its contents are the payload data carried for the application. The length of the data section is not specified in the TCP segment header. It can be calculated by subtracting the combined length of the TCP header and the encapsulating IP header from the total IP datagram length (specified in the IP header).

TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if Data Offset > 5,padded at end with "0" bytes if necessary) ...																															

- Source port (16 bits) – identifies the sending port
- Destination port (16 bits) – identifies the receiving port
- Sequence number (32 bits) – has a dual role:

  - If the `SYN` flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
  - If the `SYN` flag is clear (0), then this is the accumulated sequence number of the first data byte of this packet for the current session.
- Acknowledgment number (32 bits) – if the `ACK` flag is set then the value of this field is the next sequence number that the receiver is expecting. This acknowledges receipt of all prior bytes (if any). The first `ACK` sent by each end acknowledges the other end's initial sequence number itself, but no data.
- Data offset (4 bits) – specifies the size of the TCP header in 32-bit words. The minimum size header is 5 words and the maximum is 15 words thus giving the minimum size of 20 bytes and maximum of 60 bytes, allowing for up to 40 bytes of options in the header. This field gets its name from the fact that it is also the offset from the start of the TCP segment to the actual data.
- Reserved (3 bits) – for future use and should be set to zero
- Flags (9 bits) (aka Control bits) – contains 9 1-bit flags
  - `NS` (1 bit) – ECN-nonce concealment protection (added to header by RFC 3540).
  - `CWR` (1 bit) – Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the `ECE` flag set and had responded in congestion control mechanism (added to header by RFC 3168).
  - `ECE` (1 bit) – ECN-Echo indicates
    - If the `SYN` flag is set (1), that the TCP peer is ECN capable.
    - If the `SYN` flag is clear (0), that a packet with Congestion Experienced flag in IP header set is received during normal transmission (added to header by RFC 3168).
  - `URG` (1 bit) – indicates that the Urgent pointer field is significant
  - `ACK` (1 bit) – indicates that the Acknowledgment field is significant. All packets after the initial `SYN` packet sent by the client should have this flag set.
  - `PSH` (1 bit) – Push function. Asks to push the buffered data to the receiving application.
  - `RST` (1 bit) – Reset the connection
  - `SYN` (1 bit) – Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. Some other flags change meaning based on this flag, and some are only valid for when it is set, and others when it is clear.
  - `FIN` (1 bit) – No more data from sender
- Window size (16 bits) – the size of the *receive window*, which specifies the number of bytes (beyond the sequence number in the acknowledgment field) that the receiver is currently willing to receive (*see Flow control and Window Scaling*)
- Checksum (16 bits) – The 16-bit checksum field is used for error-checking of the header and data
- Urgent pointer (16 bits) – if the `URG` flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte
- Options (Variable 0-320 bits, divisible by 32) – The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). The Option-Kind field indicates the type of option, and is the only field that is not optional. Depending on what kind of option we are dealing with, the next two fields may be set: the Option-Length field indicates the total length of the option, and the Option-Data field contains the value of the option, if applicable. For example, an Option-Kind byte of 0x01 indicates that this is a No-Op option used only for padding, and does not have an Option-Length or Option-Data byte following it. An Option-Kind byte of 0 is the End Of Options option, and is also only one byte. An Option-Kind byte of 0x02 indicates that this is the Maximum Segment Size option, and will be followed by a byte specifying the length of the MSS field (should be 0x04). Note that this length is the total length of the given options field, including Option-Kind and Option-Length bytes. So while the MSS value is typically expressed in

two bytes, the length of the field will be 4 bytes (+2 bytes of kind and length). In short, an MSS option field with a value of 0x05B4 will show up as (0x02 0x04 0x05B4) in the TCP options section.

- Padding – The TCP header padding is used to ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros.[6]

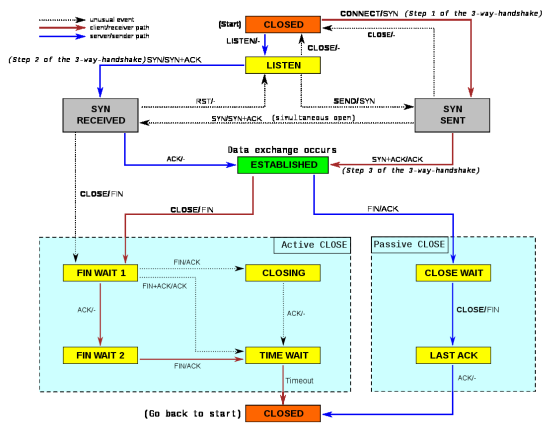
Some options may only be sent when `SYN` is set; they are indicated below as `[SYN]` Option-Kind and standard lengths given as (Option-Kind,Option-Length).

- 0 (8 bits) - End of options list
- 1 (8 bits) - No operation (NOP, Padding) This may be used to align option fields on 32-bit boundaries for better performance.
- 2,4,*SS* (32 bits) - Maximum segment size (*see maximum segment size*) `[SYN]`
- 3,3,*S* (24 bits) - Window scale (*see window scaling for details*) `[SYN]`[7]
- 4,2 (16 bits) - Selective Acknowledgement permitted. `[SYN]` (*See selective acknowledgments for details*)[8]
- 5,*N,BBBB,EEEE,...* (variable bits, *N* is either 10, 18, 26, or 34)- Selective ACKnowledgement (SACK)[9] These first two bytes are followed by a list of 1-4 blocks being selectively acknowledged, specified as 32-bit begin/end pointers.
- 8,10,*TTTT,EEEE* (80 bits)- Timestamp and echo of previous timestamp (*see TCP timestamps for details*)[10]
- 14,3,*S* (24 bits) - TCP Alternate Checksum Request. `[SYN]`[11]
- 15,*N,...* (variable bits) - TCP Alternate Checksum Data.

(The remaining options are obsolete, experimental, not yet standardized, or unassigned)

Protocol operation

TCP protocol operations may be divided into three phases. Connections must be properly established in a multi-step handshake process (*connection establishment*) before entering the *data transfer* phase. After data transmission is completed, the *connection termination* closes established virtual circuits and releases all allocated resources.



A Simplified TCP State Diagram. See TCP EFSM diagram [12] for a more detailed state diagram including the states inside the ESTABLISHED state.

A TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the *Internet socket*. During the lifetime of a TCP connection it undergoes a series of state changes:[13]

1. LISTENING : In case of a server, waiting for a connection request from any remote client.
2. SYN-SENT : waiting for the remote peer to send back a TCP segment with the SYN and ACK flags set. (usually set by TCP clients)
3. SYN-RECEIVED : waiting for the remote peer to send back an acknowledgment after having sent back a connection acknowledgment to the remote peer. (usually set by TCP servers)
4. ESTABLISHED : the port is ready to receive/send data from/to the remote peer.
5. FIN-WAIT-1 :
6. FIN-WAIT-2 :Indicates that the client is waiting for the server's fin segment (which indicates the server's application process is ready to close and the server is ready to initiate its side of the connection termination)
7. CLOSE-WAIT :

8. LAST-ACK : indicates that the server is in the process of sending its own fin segment (which indicates the server's application process is ready to close and the server is ready to initiate it's side of the connection termination)
9. TIME-WAIT : represents waiting for enough time to pass to be sure the remote peer received the acknowledgment of its connection termination request. According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes known as a MSL (maximum segment lifetime).
10. CLOSED : connection is closed

Connection establishment

To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:

1. **SYN**: The active open is performed by the client sending a SYN to the server. It sets the segment's sequence number to a random value A.
2. **SYN-ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another random number, B.
3. **ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A + 1, and the acknowledgement number is set to one more than the received sequence number i.e. B + 1.

At this point, both the client and server have received an acknowledgment of the connection.

Resource usage

Most implementations allocate an entry in a table that maps a session to a running operating system process. Because TCP packets do not include a session identifier, both endpoints identify the session using the client's address and port. Whenever a packet is received, the TCP implementation must perform a lookup on this table to find the destination process.

The number of sessions in the server side is limited only by memory and can grow as new connections arrive, but the client must allocate a random port before sending the first SYN to the server. This port remains allocated during the whole conversation, and effectively limits the number of outgoing connections from each of the client's IP addresses. If an application fails to properly close unrequired connections, a client can run out of resources and become unable to establish new TCP connections, even from other applications.

Both endpoints must also allocate space for unacknowledged packets and received (but unread) data.

Data transfer

There are a few key features that set TCP apart from User Datagram Protocol:

- Ordered data transfer - the destination host rearranges according to sequence number[2]
- Retransmission of lost packets - any cumulative stream not acknowledged is retransmitted[2]
- Error-free data transfer[14]
- Flow control - limits the rate a sender transfers data to guarantee reliable delivery. The receiver continually hints the sender on how much data can be received (controlled by the sliding window). When the receiving host's buffer fills, the next acknowledgment contains a 0 in the window size, to stop transfer and allow the data in the buffer to be processed.[2]
- Congestion control [2]

Reliable transmission

TCP uses a *sequence number* to identify each byte of data. The sequence number identifies the order of the bytes sent from each computer so that the data can be reconstructed in order, regardless of any fragmentation, disordering, or packet loss that may occur during transmission. For every payload byte transmitted, the sequence number must be incremented. In the first two steps of the 3-way handshake, both computers exchange an initial sequence number (ISN). This number can be arbitrary, and should in fact be unpredictable to defend against TCP Sequence Prediction Attacks.

TCP primarily uses a *cumulative acknowledgment* scheme, where the receiver sends an acknowledgment signifying that the receiver has received all data preceding the acknowledged sequence number. The sender sets the sequence number field to the sequence number of the first payload byte in the segment's data field, and the receiver sends an acknowledgment specifying the sequence number of the next byte they expect to receive. For example, if a sending computer sends a packet containing four payload bytes with a sequence number field of 100, then the sequence numbers of the four payload bytes are 100, 101, 102 and 103. When this packet arrives at the receiving computer, it would send back an acknowledgment number of 104 since that is the sequence number of the next byte it expects to receive in the next packet.

In addition to cumulative acknowledgments, TCP receivers can also send selective acknowledgments to provide further information.

If the sender infers that data has been lost in the network, it retransmits the data.

Error detection

Sequence numbers and acknowledgments cover discarding duplicate packets, retransmission of lost packets, and ordered-data transfer. To assure correctness a checksum field is included (*see TCP segment structure for details on checksumming*).

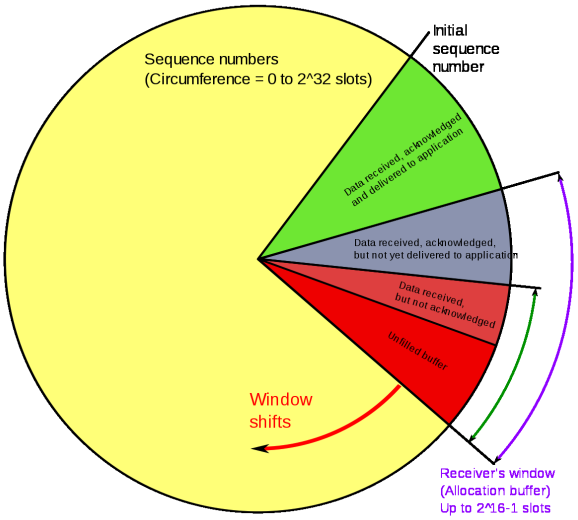
The TCP checksum is a weak check by modern standards. Data Link Layers with high bit error rates may require additional link error correction/detection capabilities. The weak checksum is partially compensated for by the common use of a CRC or better integrity check at layer 2, below both TCP and IP, such as is used in PPP or the Ethernet frame. However, this does not mean that the 16-bit TCP checksum is redundant: remarkably, introduction of errors in packets between CRC-protected hops is common, but the end-to-end 16-bit TCP checksum catches most of these simple errors.[15] This is the end-to-end principle at work.

Flow control

TCP uses an end-to-end flow control protocol to avoid having the sender send data too fast for the TCP receiver to receive and process it reliably. Having a mechanism for flow control is essential in an environment where machines of diverse network speeds communicate. For example, if a PC sends data to a hand-held PDA that is slowly processing received data, the PDA must regulate data flow so as not to be overwhelmed.[2]

TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the *receive window* field the amount of additional received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgment and window update from the receiving host.

When a receiver advertises a window size of 0, the sender stops sending data and starts the *persist timer*. The persist timer is used to protect TCP from a deadlock situation that could arise if a subsequent window size update from the receiver is lost, and the sender cannot send more data until receiving a new window size update from the receiver. When the persist timer expires, the TCP sender attempts recovery by sending a small packet so that the receiver responds by sending another acknowledgement containing the new window size.



TCP sequence numbers and receive windows behave very much like a clock. The receive window shifts each time the receiver receives and acknowledges a new segment of data. Once it runs out of sequence numbers, the sequence number loops back to 0.

If a receiver is processing incoming data in small increments, it may repeatedly advertise a small receive window. This is referred to as the silly window syndrome, since it is inefficient to send only a few bytes of data in a TCP segment, given the relatively large overhead of the TCP header. TCP senders and receivers typically employ flow control logic to specifically avoid repeatedly sending small segments. The sender-side silly window syndrome avoidance logic is referred to as Nagle's algorithm.

Congestion control

The final main aspect of TCP is congestion control. TCP uses a number of mechanisms to achieve high performance and avoid congestion collapse, where network performance can fall by several orders of magnitude. These mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger collapse. They also yield an approximately max-min fair allocation between flows.

Acknowledgments for data sent, or lack of acknowledgments, are used by senders to infer network conditions between the TCP sender and receiver. Coupled with timers, TCP senders and receivers can alter the behavior of the flow of data. This is more generally referred to as congestion control and/or network congestion avoidance.

Modern implementations of TCP contain four intertwined algorithms: Slow-start, congestion avoidance, fast retransmit, and fast recovery (RFC 5681).

In addition, senders employ a *retransmission timeout* (RTO) that is based on the estimated round-trip time (or RTT) between the sender and receiver, as well as the variance in this round trip time. The behavior of this timer is specified in RFC 2988. There are subtleties in the estimation of RTT. For example, senders must be careful when calculating RTT samples for retransmitted packets; typically they use Karn's Algorithm or TCP timestamps (see RFC 1323). These individual RTT samples are then averaged over time to create a Smoothed Round Trip Time (SRTT) using Jacobson's algorithm. This SRTT value is what is finally used as the round-trip time estimate.

Enhancing TCP to reliably handle loss, minimize errors, manage congestion and go fast in very high-speed environments are ongoing areas of research and standards development. As a result, there are a number of TCP congestion avoidance algorithm variations.

Maximum segment size

The maximum segment size (MSS) is the largest amount of data, specified in bytes, that TCP is willing to receive in a single segment. For best performance, the MSS should be set small enough to avoid IP fragmentation, which can lead to packet loss and excessive retransmissions. To try to accomplish this, typically the MSS is announced by each side using the MSS option when the TCP connection is established, in which case it is derived from the maximum transmission unit (MTU) size of the data link layer of the networks to which the sender and receiver are directly attached. Furthermore, TCP senders can use path MTU discovery to infer the minimum MTU along the network path between the sender and receiver, and use this to dynamically adjust the MSS to avoid IP fragmentation within the network.

MSS announcement is also often called "MSS negotiation". Strictly speaking, the MSS is not "negotiated" between the originator and the receiver, because that would imply that both originator and receiver will negotiate and agree upon a single, unified MSS that applies to all communication in both directions of the connection. In fact, two completely independent values of MSS are permitted for the two directions of data flow in a TCP connection.[16] This situation may arise, for example, if one of the devices participating in a connection has an extremely limited amount memory reserved (perhaps even smaller than the overall discovered Path MTU) for processing incoming TCP segments.

Selective acknowledgments

Relying purely on the cumulative acknowledgment scheme employed by the original TCP protocol can lead to inefficiencies when packets are lost. For example, suppose 10,000 bytes are sent in 10 different TCP packets, and the first packet is lost during transmission. In a pure cumulative acknowledgment protocol, the receiver cannot say that it received bytes 1,000 to 9,999 successfully, but failed to receive the first packet, containing bytes 0 to 999. Thus the sender may then have to resend all 10,000 bytes.

To solve this problem TCP employs the *selective acknowledgment (SACK)* option, defined in RFC 2018, which allows the receiver to acknowledge discontinuous blocks of packets that were received correctly, in addition to the sequence number of the last contiguous byte received successively, as in the basic TCP acknowledgment. The acknowledgement can specify a number of *SACK blocks*, where each SACK block is conveyed by the starting and ending sequence numbers of a contiguous range that the receiver correctly received. In the example above, the receiver would send SACK with sequence numbers 1000 and 9999. The sender thus retransmits only the first packet, bytes 0 to 999.

An extension to the SACK option is the duplicate-SACK option, defined in RFC 2883. An out-of-order packet delivery can often falsely indicate the TCP sender of lost packet and, in turn, the TCP sender retransmits the suspected-to-be-lost packet and slow down the data delivery to prevent network congestion. The TCP sender undoes the action of slow-down, that is a recovery of the original pace of data transmission, upon receiving a D-SACK that indicates the retransmitted packet is duplicate.

The SACK option is not mandatory and it is used only if both parties support it. This is negotiated when connection is established. SACK uses the optional part of the TCP header (*see TCP segment structure for details*). The use of SACK is widespread - all popular TCP stacks support it. Selective acknowledgment is also used in Stream Control Transmission Protocol (SCTP).

Window scaling

For more efficient use of high bandwidth networks, a larger TCP window size may be used. The TCP window size field controls the flow of data and its value is limited to between 2 and 65,535 bytes.

Since the size field cannot be expanded, a scaling factor is used. The TCP window scale option, as defined in RFC 1323, is an option used to increase the maximum window size from 65,535 bytes to 1 gigabyte. Scaling up to larger window sizes is a part of what is necessary for TCP Tuning.

The window scale option is used only during the TCP 3-way handshake. The window scale value represents the number of bits to left-shift the 16-bit window size field. The window scale value can be set from 0 (no shift) to 14 for each direction independently. Both sides must send the option in their SYN segments to enable window scaling in either direction.

Some routers and packet firewalls rewrite the window scaling factor during a transmission. This causes sending and receiving sides to assume different TCP window sizes. The result is non-stable traffic that may be very slow. The problem is visible on some sending and receiving sites behind the path of defective routers.[17]

TCP timestamps

TCP timestamps, defined in RFC 1323, can help TCP determine in which order packets were sent. TCP timestamps are not normally aligned to the system clock and start at some random value. Many operating systems will increment the timestamp for every elapsed milisecond; however the RFC only states that the tics should be proportional.

There are two timestamp fields:

```
a 4-byte sender timestamp value (my timestamp)
a 4-byte echo reply timestamp value (the most recent timestamp received from you).
```

TCP timestamps are used in an algorithm known as *Protection Against Wrapped Sequence* numbers, or *PAWS* (see RFC 1323 for details). PAWS is used when the TCP window size exceeds the possible numbers of sequence numbers (2^32 or 4 billion/gig). In the case where a packet was potentially retransmitted it answers the question: "Is this sequence number in the first 4GB or the second?" And the timestamp is used to break the tie.

RFC 1323 incorrectly states in section 2.2 that the window scale must be limited to 2^14 to remain under 1GB (which is correct, but the sequence number limit is 4GB); however a scale of 16 and a window size of 65535 would be 65536 less than the 2^32 possible sequence numbers and thus an acceptable yet excessive value. Because of this error many systems have limited the max scale to 2^14 to "follow the RFC".

Also, the Eifel detection algorithm (RFC 3522) uses TCP timestamps to determine if retransmissions are occurring because packets are lost or simply out of order.

Out of band data

One is able to interrupt or abort the queued stream instead of waiting for the stream to finish. This is done by specifying the data as *urgent*. This tells the receiving program to process it immediately, along with the rest of the urgent data. When finished, TCP informs the application and resumes back to the stream queue. An example is when TCP is used for a remote login session, the user can send a keyboard sequence that interrupts or aborts the program at the other end. These signals are most often needed when a program on the remote machine fails to operate correctly. The signals must be sent without waiting for the program to finish its current transfer.[2]

TCP OOB data was not designed for the modern Internet. The *urgent* pointer only alters the processing on the remote host and doesn't expedite any processing on the network itself. When it gets to the remote host there are two slightly different interpretations of the protocol, which means only single bytes of OOB data are reliable. This is assuming it's reliable at all as it's one of the least commonly used protocol elements and tends to be poorly implemented. [18] [19]

Forcing data delivery

Normally, TCP waits for 200 ms or for a full packet of data to send (Nagle's Algorithm = tries to group small messages into a single packet). This creates minor, but potentially serious delays if repeated constantly during a file transfer. For example a typical send block would be 4KB, a typical MSS is 1460, so 2 packets go out on a 10Mb/s ethernet taking ~1.2 ms each followed by a third carrying the remaining 1176 after a 197ms pause because TCP is waiting for a full buffer.

In the case of telnet, each user keystroke is echoed back by the server before the user can see it on the screen. This delay would become very annoying.

Setting the socket option `TCP_NODELAY` overrides the default 200ms send delay. Application programs use this socket option to force output to be sent after writing a character or line of characters.

The RFC defines the `PSH` push bit as "a message to the receiving TCP stack to send this data immediately up to the receiving application".[2] There is no way to indicate or control it in User space using Berkeley sockets and it is controlled by Protocol stack only.[20]

Connection termination

The connection termination phase uses, at most, a four-way handshake, with each side of the connection terminating independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. Therefore, a typical tear-down requires a pair of FIN and ACK segments from each TCP endpoint. After both FIN/ACK exchanges are concluded, the terminating side waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections; this prevents confusion due to delayed packets being delivered during subsequent connections.

A connection can be "half-open", in which case one side has terminated its end, but the other has not. The side that has terminated can no longer send any data into the connection, but the other side can. The terminating side should continue reading the data until the other side terminates as well.

It is also possible to terminate the connection by a 3-way handshake, when host A sends a FIN and host B replies with a FIN & ACK (merely combines 2 steps into one) and host A replies with an ACK.[21] This is perhaps the most common method.

It is possible for both hosts to send FINs simultaneously then both just have to ACK. This could possibly be considered a 2-way handshake since the FIN/ACK sequence is done in parallel for both directions.

Some host TCP stacks may implement a half-duplex close sequence, as Linux or HP-UX do. If such a host actively closes a connection but still has not read all the incoming data the stack already received from the link, this host sends a RST instead of a FIN (Section 4.2.2.13 in RFC 1122 [22]). This allows a TCP application to be sure the remote application has read all the data the former sent—waiting the FIN from the remote side, when it actively closes the connection. However, the remote TCP stack cannot distinguish between a *Connection Aborting RST* and this *Data Loss RST*. Both cause the remote stack to throw away all the data it received, but that the application still didn't read.

Some application protocols may violate the OSI model layers, using the TCP open/close handshaking for the application protocol open/close handshaking - these may find the RST problem on active close. As an example:

```
s = connect(remote);
send(s, data);
close(s);
```

For a usual program flow like above, a TCP/IP stack like that described above does not guarantee that all the data arrives to the other application.

Vulnerabilities

TCP may be attacked in a variety of ways. The results of a thorough security assessment of TCP, along with possible mitigations for the identified issues, was published in 2009,[23] and is currently being pursued within the IETF.[24]

Denial of service

By using a spoofed IP address and repeatedly sending purposely assembled SYN packets, attackers can cause the server to consume large amounts of resources keeping track of the bogus connections. This is known as a SYN flood attack. Proposed solutions to this problem include SYN cookies and Cryptographic puzzles. Sockstress is a similar attack, that might be mitigated with system resource management.[25] An advanced DoS attack involving the exploitation of the TCP Persist Timer was analyzed at Phrack #66.[26]

Connection hijacking

An attacker who is able to eavesdrop a TCP session and redirect packets can hijack a TCP connection. To do so, the attacker learns the sequence number from the ongoing communication and forges a false segment that looks like the next segment in the stream. Such a simple hijack can result in one packet being erroneously accepted at one end. When the receiving host acknowledges the extra segment to the other side of the connection, synchronization is lost. Hijacking might be combined with ARP or routing attacks that allow taking control of the packet flow, so as to get permanent control of the hijacked TCP connection.[27]

Impersonating a different IP address was not difficult prior to RFC 1948, when the initial *sequence number* was easily guessable. That allowed an attacker to blindly send a sequence of packets that the receiver would believe to come from a different IP address, without the need to deploy ARP or routing attacks: it is enough to ensure that the legitimate host of the impersonated IP address is down, or bring it to that condition using denial of service attacks. This is why the initial sequence number is chosen at random.

TCP ports

TCP uses port numbers to identify sending and receiving application end-points on a host, or *Internet sockets*. Each side of a TCP connection has an associated 16-bit unsigned port number (0-65535) reserved by the sending or receiving application. Arriving TCP data packets are identified as belonging to a specific TCP connection by its sockets, that is, the combination of source host address, source port, destination host address, and destination port. This means that a server computer can provide several clients with several services simultaneously, as long as a client takes care of initiating any simultaneous connections to one destination port from different source ports.

Port numbers are categorized into three basic categories: well-known, registered, and dynamic/private. The well-known ports are assigned by the Internet Assigned Numbers Authority (IANA) and are typically used by system-level or root processes. Well-known applications running as servers and passively listening for connections typically use these ports. Some examples include: FTP (20 and 21), SSH (22), TELNET (23), SMTP (25) and HTTP (80). Registered ports are typically used by end user applications as ephemeral source ports when contacting servers, but they can also identify named services that have been registered by a third party. Dynamic/private ports can also be used by end user applications, but are less commonly so. Dynamic/private ports do not contain any meaning outside of any particular TCP connection.

Development

TCP is a complex protocol. However, while significant enhancements have been made and proposed over the years, its most basic operation has not changed significantly since its first specification RFC 675 in 1974, and the v4 specification RFC 793, published in September 1981. RFC 1122, Host Requirements for Internet Hosts, clarified a number of TCP protocol implementation requirements. RFC 2581, TCP Congestion Control, one of the most important TCP-related RFCs in recent years, describes updated algorithms that avoid undue congestion. In 2001, RFC 3168 was written to describe explicit congestion notification (ECN), a congestion avoidance signaling mechanism.

The original TCP congestion avoidance algorithm was known as "TCP Tahoe", but many alternative algorithms have since been proposed (including TCP Reno, TCP Vegas, FAST TCP, TCP New Reno, and TCP Hybla).

TCP Interactive (iTCP) [28] is a research effort into TCP extensions that allows applications to subscribe to TCP events and register handler components that can launch applications for various purposes, including application-assisted congestion control.

Multipath TCP (MPTCP) [29] [30] is an ongoing effort within the IETF that aims at allowing a TCP connection to use multiple paths to maximise resource usage and increase redundancy. The redundancy offered by Multipath TCP in the context of wireless networks [31] enables statistical multiplexing of resources, and thus increases TCP throughput dramatically. Multipath TCP also brings performance benefits in datacenter environments [32] and an implementation in the Linux kernel is being developed.[33]

TCP Cookie Transactions (TCPCT) is an extension proposed in December 2009 to secure servers against denial-of-service attacks. Unlike SYN cookies, TCPCT does not conflict with other TCP extensions such as window scaling. TCPCT was designed due to necessities of DNSSEC, where servers have to handle large numbers of short-lived TCP connections.

tcpcrypt is an extension proposed in July 2010 to provide transport-level encryption directly in TCP itself. It's designed to work transparently and not require any configuration. Unlike TLS (SSL), tcpcrypt itself does not provide authentication, but provides simple primitives down to the application to do that. As of 2010, the first tcpcrypt IETF draft has been published and implementations exist for several major platforms.

TCP over wireless networks

TCP has been optimized for wired networks. Any packet loss is considered to be the result of network congestion and the congestion window size is reduced dramatically as a precaution. However, wireless links are known to experience sporadic and usually temporary losses due to fading, shadowing, hand off, and other radio effects, that cannot be considered congestion. After the (erroneous) back-off of the congestion window size, due to wireless packet loss, there can be a congestion avoidance phase with a conservative decrease in window size. This causes the radio link to be underutilized. Extensive research has been done on the subject of how to combat these harmful effects. Suggested solutions can be categorized as end-to-end solutions (which require modifications at the client or server),[34] link layer solutions (such as RLP in cellular networks), or proxy based solutions (which require some changes in the network without modifying end nodes.[34] [35]

Hardware implementations

One way to overcome the processing power requirements of TCP is to build hardware implementations of it, widely known as TCP Offload Engines (TOE). The main problem of TOEs is that they are hard to integrate into computing systems, requiring extensive changes in the operating system of the computer or device. One company to develop such a device was Alacritech.

Debugging

A packet sniffer, which intercepts TCP traffic on a network link, can be useful in debugging networks, network stacks and applications that use TCP by showing the user what packets are passing through a link. Some networking stacks support the SO_DEBUG socket option, which can be enabled on the socket using setsockopt. That option dumps all the packets, TCP states, and events on that socket, which is helpful in debugging. Netstat is another utility that can be used for debugging.

Alternatives

For many applications TCP is not appropriate. One big problem (at least with normal implementations) is that the application cannot get at the packets coming after a lost packet until the retransmitted copy of the lost packet is received. This causes problems for real-time applications such as streaming media, real-time multiplayer games and voice over IP (VoIP) where it is sometimes more useful to get most of the data in a timely fashion than it is to get all of the data in order.

For both historical and performance reasons, most storage area networks (SANs) prefer to use Fibre Channel protocol (FCP) instead of TCP/IP.

Also for embedded systems, network booting and servers that serve simple requests from huge numbers of clients (e.g. DNS servers) the complexity of TCP can be a problem. Finally, some tricks such as transmitting data between two hosts that are both behind NAT (using STUN or similar systems) are far simpler without a relatively complex protocol like TCP in the way.

Generally, where TCP is unsuitable, the User Datagram Protocol (UDP) is used. This provides the application multiplexing and checksums that TCP does, but does not handle building streams or retransmission, giving the application developer the ability to code them in a way suitable for the situation, or to replace them with other methods like forward error correction or interpolation.

SCTP is another IP protocol that provides reliable stream oriented services similar to TCP. It is newer and considerably more complex than TCP, and has not yet seen widespread deployment. However, it is especially designed to be used in situations where reliability and near-real-time considerations are important.

Venturi Transport Protocol (VTP) is a patented proprietary protocol that is designed to replace TCP transparently to overcome perceived inefficiencies related to wireless data transport.

TCP also has issues in high bandwidth environments. The TCP congestion avoidance algorithm works very well for ad-hoc environments where the data sender is not known in advance, but if the environment is predictable, a timing based protocol such as Asynchronous Transfer Mode (ATM) can avoid TCP's retransmits overhead.

Multipurpose Transaction Protocol (MTP/IP) is patented proprietary software that is designed to adaptively achieve high throughput and transaction performance in a wide variety of network conditions, particularly those where TCP is perceived to be inefficient.

Checksum computation

TCP checksum for IPv4

When TCP runs over IPv4, the method used to compute the checksum is defined in RFC 793:

> *The checksum field is the 16 bit one's complement of the one's complement sum of all 16-bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.*

In other words, after appropriate padding, all 16-bit words are added using one's complement arithmetic. The sum is then bitwise complemented and inserted as the checksum field. A pseudo-header that mimics the IPv4 packet header used in the checksum computation is shown in the table below.

TCP pseudo-header for checksum computation (IPv4)

Bit offset	0–3	4–7	8–15	16–31
0	Source address			
32	Destination address			
64	Zeros		Protocol	TCP length
96	Source port			Destination port
128	Sequence number			
160	Acknowledgement number			
192	Data offset	Reserved	Flags	Window
224	Checksum			Urgent pointer
256	Options (optional)			
256/288+	Data			

The source and destination addresses are those of the IPv4 header. The protocol value is 6 for TCP (cf. List of IP protocol numbers). The TCP length field is the length of the TCP header and data.

TCP checksum for IPv6

When TCP runs over IPv6, the method used to compute the checksum is changed, as per RFC 2460:

> *Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses.*

A pseudo-header that mimics the IPv6 header for computation of the checksum is shown below.

TCP pseudo-header for checksum computation (IPv6)

Bit offset	0 - 7	8–15	16–23	24–31
0	Source address			
32				
64				
96				
128	Destination address			
160				
192				
224				
256	TCP length			
288	Zeros			Next header
320	Source port		Destination port	
352	Sequence number			
384	Acknowledgement number			
416	Data offset	Reserved	Flags	Window
448	Checksum		Urgent pointer	
480	Options (optional)			
480/512+	Data			

- Source address – the one in the IPv6 header
- Destination address – the final destination; if the IPv6 packet doesn't contain a Routing header, TCP uses the destination address in the IPv6 header, otherwise, at the originating node, it uses the address in the last element of the Routing header, and, at the receiving node, it uses the destination address in the IPv6 header.
- TCP length – the length of the TCP header and data
- Next Header – the protocol value for TCP

Checksum offload

Many TCP/IP software stack implementations provide options to use hardware assistance to automatically compute the checksum in the network adapter prior to transmission onto the network or upon reception from the network for validation. This may relieve OS to use precious CPU cycles calculating checksum. Hence, overall network performance is increased.

This feature may cause packet analyzers detecting outbound network traffic upstream of the network adapter and unaware or uncertain about the use of checksum offload to report invalid checksum in outbound packets.

References

[1] Vinton G. Cerf, Robert E. Kahn, *A Protocol for Packet Network Intercommunication*, IEEE Transactions on Communications, Vol. 22, No. 5, May 1974 pp. 637-648

[2] Comer, Douglas E. (2006). *Internetworking with TCP/IP:Principles, Protocols, and Architecture*. **1** (5th ed.). Prentice Hall. ISBN 0131876716.

[3] TCP (Linktionary term) (http://www.linktionary.com/t/tcp.html)

[4] RFC 791 - section 2.1 (http://tools.ietf.org/html/rfc791#section-2.1)

[5] RFC 793 (http://tools.ietf.org/html/rfc793)

[6] RFC 793 section 3.1

[7] RFC 1323, TCP Extensions for High Performance, Section 2.2 (http://tools.ietf.org/html/rfc1323#page-9)

[8] RFC 2018, TCP Selective Acknowledgement Options, Section 2 (http://tools.ietf.org/html/rfc2018#section-2)

[9] RFC 2018, TCP Selective Acknowledgement Options, Section 3 (http://tools.ietf.org/html/rfc2018#section-3)

[10] RFC 1323, TCP Extensions for High Performance, Section 3.2 (http://tools.ietf.org/html/rfc1323#page-11)

[11] RFC 1146, TCP Alternate Checksum Options (http://tools.ietf.org/html/rfc1146#page-2)

[12] http://www.medianet.kent.edu/techreports/TR2005-07-22-tcp-EFSM.pdf

[13] RFC 793 Section 3.2

[14] "TCP Definition" (http://www.linfo.org/tcp.html). . Retrieved 2011-03-12.

[15] Stone; Partridge (2000). "When The CRC and TCP Checksum Disagree" (http://citeseer.ist.psu.edu/stone00when.html). *Sigcomm*.

[16] RFC 879 (http://www.faqs.org/rfcs/rfc879.html)

[17] http://lwn.net/Articles/92727/

[18] Gont, Fernando (2008-11). "On the implementation of TCP urgent data" (http://www.gont.com.ar/talks/IETF73/ietf73-tcpm-urgent-data.ppt). 73rd IETF meeting. . Retrieved 2009-01-04.

[19] Peterson, Larry (2003). *Computer Networks*. Morgan Kaufmann. pp. 401. ISBN 155860832X.

[20] Richard W. Stevens (2006). *TCP/IP Illustrated. Vol. 1, The protocols* (http://books.google.com/books?id=b2elQwAACAAJ). Addison-Wesley. pp. Chapter 20. ISBN 978-0-201-63346-7. . Retrieved 14 November 2011.

[21] Tanenbaum, Andrew S. (2003-03-17). *Computer Networks* (Fourth ed.). Prentice Hall. ISBN 0-13-066102-3.

[22] http://tools.ietf.org/html/rfc1122

[23] Security Assessment of the Transmission Control Protocol (TCP) (http://www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf)

[24] Security Assessment of the Transmission Control Protocol (TCP) (http://tools.ietf.org/html/draft-ietf-tcpm-tcp-security)

[25] http://www.gont.com.ar/talks/hacklu2009/fgont-hacklu2009-tcp-security.pdf Some insights about the recent TCP DoS (Denial of Service) vulnerabilities

[26] Exploiting TCP and the Persist Timer Infiniteness (http://phrack.org/issues.html?issue=66&id=9#article)

[27] Laurent Joncheray, *Simple Active Attack Against TCP*, 1995 (http://www.usenix.org/publications/library/proceedings/security95/joncheray.html)

[28] TCP Interactive (iTCP) (http://www.medianet.kent.edu/projects_files/projectITCP.html)

[29] draft-ietf-mptcp-architecture (http://tools.ietf.org/html/draft-ietf-mptcp-architecture-03)

[30] http://datatracker.ietf.org/doc/draft-ietf-mptcp-multiaddressed/ (http://tools.ietf.org/html/draft-ietf-mptcp-multiaddressed-03)

[31] http://portal.acm.org/citation.cfm?id=1794199

[32] Raiciu; Barre; Pluntke; Greenhalgh; Wischik; Handley (2011). "Improving datacenter performance and robustness with multipath TCP" (http://inl.info.ucl.ac.be/publications/improving-datacenter-performance-and-robustness-multipath-tcp). *Sigcomm*.

[33] Barre; Paasch; Bonaventure (2011). "MultiPath TCP: From Theory to Practice" (http://inl.info.ucl.ac.be/publications/multipath-tcp-theory-practice). *IFIP Networking*.

[34] "TCP performance over CDMA2000 RLP" (http://academic.research.microsoft.com/Paper/3352358.aspx). . Retrieved 2010-08-30

[35] Muhammad Adeel & Ahmad Ali Iqbal (2004). "TCP Congestion Window Optimization for CDMA2000 Packet Data Networks" (http://www.computer.org/portal/web/csdl/doi/10.1109/ITNG.2007.190). *International Conference on Information Technology (ITNG'07)*: 31–35. doi:10.1109/ITNG.2007.190. ISBN 978-0-7695-2776-5. .

Further reading

- W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. ISBN 0-201-63346-9
- W. Richard Stevens and Gary R. Wright. TCP/IP Illustrated, Volume 2: The Implementation. ISBN 0-201-63354-X
- W. Richard Stevens. TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols. ISBN 0-201-63495-3

External links

RFC

- RFC 675 - Specification of Internet Transmission Control Program, December 1974 Version
- RFC 793 - TCP v4
- RFC 1122 - includes some error corrections for TCP
- RFC 1323 - TCP-Extensions
- RFC 1379 - Extending TCP for Transactions—Concepts
- RFC 1948 - Defending Against Sequence Number Attacks
- RFC 2018 - TCP Selective Acknowledgment Options
- RFC 2988 - Computing TCP's Retransmission Timer
- RFC 4614 - A Roadmap for TCP Specification Documents
- RFC 5681 - TCP Congestion Control

Others

- Oral history interview with Robert E. Kahn (http://www.cbi.umn.edu/oh/display.phtml?id=119), Charles Babbage Institute, University of Minnesota, Minneapolis. Focuses on Kahn's role in the development of computer networking from 1967 through the early 1980s. Beginning with his work at Bolt Beranek and Newman (BBN), Kahn discusses his involvement as the ARPANET proposal was being written, his decision to become active in its implementation, and his role in the public demonstration of the ARPANET. The interview continues into Kahn's involvement with networking when he moves to IPTO in 1972, where he was responsible for the administrative and technical evolution of the ARPANET, including programs in packet radio, the development of a new network protocol (TCP/IP), and the switch to TCP/IP to connect multiple networks.
- IANA Port Assignments (http://www.iana.org/assignments/port-numbers)
- John Kristoff's Overview of TCP (Fundamental concepts behind TCP and how it is used to transport data between two endpoints) (http://condor.depaul.edu/~jkristof/technotes/tcp.html)
- TCP fast retransmit simulation animated: slow start, sliding window, duplicated Ack, congestion window (http://www.visualland.net/tcp_histrory.php?simu=tcp_fast_retransmit&protocol=TCP&title=4.Fast transmit&ctype=1)
- TCP, Transmission Control Protocol (http://www.networksorcery.com/enp/protocol/tcp.htm)
- Checksum example (http://mathforum.org/library/drmath/view/54379.html)
- Engineer Francesco Buffa's page about Transmission Control Protocol (http://www.ilmondodelletelecomunicazioni.it/english/telematics/protocols.html)
- TCP tutorial (http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html)
- Linktionary on TCP segments (http://www.linktionary.com/s/segment_tcp.html)
- TCP Sliding Window simulation animated (ns2) (http://www.visualland.net/tcp_histrory.php?simu=tcp_swnd&protocol=TCP&title=2.Sliding Window&ctype=1)
- Multipath TCP in Linux kernel (http://inl.info.ucl.ac.be/mptcp)

Transport layer

In computer networking, the **transport layer** or **layer 4** provides end-to-end communication services for applications[1] within a layered architecture of network components and protocols. The transport layer provides convenient services such as connection-oriented data stream support, reliability, flow control, and multiplexing.

Transport layers are contained in both the TCP/IP model (RFC 1122),[2] which is the foundation of the Internet, and the Open Systems Interconnection (OSI) model of general networking. The definitions of the transport layer are slightly different in these two models. This article primarily refers to the TCP/IP model, in which TCP is largely for a convenient application programming interface to internet hosts, as opposed to the OSI-model definition of the transport layer.

The most well-known transport protocol is the Transmission Control Protocol (TCP). It lent its name to the title of the entire Internet Protocol Suite, *TCP/IP*. It is used for connection-oriented transmissions, whereas the connectionless User Datagram Protocol (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its stateful design incorporating reliable transmission and data stream services. Other prominent protocols in this group are the Datagram Congestion Control Protocol (DCCP) and the Stream Control Transmission Protocol (SCTP).

Services

There are many services that can be optionally provided by a transport-layer protocol, and different protocols may or may not implement them.

- Connection-oriented communication: Interpreting the connection as a data stream can provide many benefits to applications. It is normally easier to deal with than the underlying connection-less models, such as the Transmission Control Protocol's underlying Internet Protocol model of datagrams.
- Byte orientation: Rather than processing the messages in the underlying communication system format, it is often easier for an application to process the data stream as a sequence of bytes. This simplification helps applications work with various underlying message formats.
- Same order delivery: The network layer doesn't generally guarantee that packets of data will arrive in the same order that they were sent, but often this is a desirable feature. This is usually done through the use of segment numbering, with the receiver passing them to the application in order. This can cause head-of-line blocking.
- Reliability: Packets may be lost during transport due to network congestion and errors. By means of an error detection code, such as a checksum, the transport protocol may check that the data is not corrupted, and verify correct receipt by sending an ACK or NACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data.
- Flow control: The rate of data transmission between two nodes must sometimes be managed to prevent a fast sender from transmitting more data than can be supported by the receiving data buffer, causing a buffer overrun. This can also be used to improve efficiency by reducing buffer underrun.
- Congestion avoidance: Congestion control can control traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. For example, automatic repeat requests may keep the network in a congested state; this situation can be avoided by adding congestion avoidance to the flow control, including slow-start. This keeps the bandwidth consumption at a low level in the beginning of the transmission, or after packet retransmission.
- Multiplexing: Ports can provide multiple endpoints on a single node. For example, the name on a postal address is a kind of multiplexing, and distinguishes between different recipients of the same location. Computer applications will each listen for information on their own ports, which enables the use of more than one network service at the same time. It is part of the transport layer in the TCP/IP model, but of the session layer in the OSI model.

Analysis

The transport layer is responsible for delivering data to the appropriate application process on the host computers. This involves statistical multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each transport-layer data packet. Together with the source and destination IP address, the port numbers constitutes a network socket, i.e. an identification address of the process-to-process communication. In the OSI model, this function is supported by the session layer.

Some transport-layer protocols, for example TCP, but not UDP, support virtual circuits, i.e. provide connection oriented communication over an underlying packet oriented datagram network. A byte-stream is delivered while hiding the packet mode communication for the application processes. This involves connection establishment, dividing of the data stream into packets called segments, segment numbering and reordering of out-of order data.

Finally, some transport-layer protocols, for example TCP, but not UDP, provide end-to-end reliable communication, i.e. error recovery by means of error detecting code and automatic repeat request (ARQ) protocol. The ARQ protocol also provides flow control, which may be combined with congestion avoidance.

UDP is a very simple protocol, and does not provide virtual circuits, nor reliable communication, delegating these functions to the application program. UDP packets are called datagrams, rather than segments.

TCP is used for many protocols, including HTTP web browsing and email transfer. UDP may be used for multicasting and broadcasting, since retransmissions are not possible to a large amount of hosts. UDP typically gives higher throughput and shorter latency, and is therefore often used for real-time multimedia communication where packet loss occasionally can be accepted, for example IP-TV and IP-telephony, and for online computer games.

In many non-IP-based networks, for example X.25, Frame Relay and ATM, the connection oriented communication is implemented at network layer or data link layer rather than the transport layer. In X.25, in telephone network modems and in wireless communication systems, reliable node-to-node communication is implemented at lower protocol layers.

The OSI model defines five classes of transport protocols: *TP0*, providing the least error recovery, to *TP4*, which is designed for less reliable networks.

Protocols

The exact definition of what qualifies as a transport-layer protocol is not firm. The following is a short list:

- ATP, AppleTalk Transaction Protocol
- CUDP, Cyclic UDP
- DCCP, Datagram Congestion Control Protocol
- FCP, Fiber Channel Protocol
- IL, IL Protocol
- NBF, NetBIOS Frames protocol
- RDP, Reliable Datagram Protocol
- SCTP, Stream Control Transmission Protocol
- SPX, Sequenced Packet Exchange
- SST, Structured Stream Transport
- TCP, Transmission Control Protocol
- UDP, User Datagram Protocol
- UDP Lite
- µTP, Micro Transport Protocol

Comparison of transport-layer protocols

Feature Name	UDP	UDP Lite	TCP	SCTP	DCCP	RUDP
Packet header size	8 Bytes	8 Bytes	20-60 Bytes	12 Bytes	12 or 16 bytes	
Transport-layer packet entity	Datagram	Datagram	Segment	Datagram	Datagram	Datagram
Connection oriented	No	No	Yes	Yes	Yes	No
Reliable transport	No	No	Yes	Yes	No	Yes
Unreliable transport	Yes	Yes	No	Yes	Yes	Yes
Preserve message boundary	Yes	Yes	No	Yes	Yes	Unsure
Ordered delivery	No	No	Yes	Yes	No	No
Unordered delivery	Yes	Yes	No	Yes	Yes	Yes
Data checksum	Optional	Yes	Yes	Yes	Yes	Unsure
Checksum size (bits)	16	16	16	32	16	Unsure
Partial checksum	No	Yes	No	No	Yes	No
Path MTU	No	No	Yes	Yes	Yes	Unsure
Flow control	No	No	Yes	Yes	No	
Congestion control	No	No	Yes	Yes	Yes	Unsure
ECN support	No	No	Yes	Yes	Yes	
Multiple streams	No	No	No	Yes	No	No
Multi-homing support	No	No	No	Yes	No	No
Bundling / Nagle	No	No	Yes	Yes	No	Unsure
NAT friendly[3]	Yes	Yes	Yes	No	Yes	Yes

Comparison of OSI transport protocols

The OSI model defines five classes of connection-mode transport protocols designated class 0 (TP0) to class 4 (TP4). Class 0 contains no error recovery, and was designed for use on network layers that provide error-free connections. Class 4 is closest to TCP, although TCP contains functions, such as the graceful close, which OSI assigns to the session layer. All OSI connection-mode protocol classes provide expedited data and preservation of record boundaries. Detailed characteristics of the classes are shown in the following table:[4]

Service	TP0	TP1	TP2	TP3	TP4
Connection oriented network	Yes	Yes	Yes	Yes	Yes
Connectionless network	No	No	No	No	Yes
Concatenation and separation	No	Yes	Yes	Yes	Yes
Segmentation and reassembly	Yes	Yes	Yes	Yes	Yes
Error Recovery	No	Yes	No	Yes	Yes
Reinitiate connection (if an excessive number of PDUs are unacknowledged)	No	Yes	No	Yes	No
multiplexing and demultiplexing over a single virtual circuit	No	No	Yes	Yes	Yes
Explicit flow control	No	No	Yes	Yes	Yes
Retransmission on timeout	No	No	No	No	Yes

Reliable Transport Service	No	Yes	No	Yes	Yes

References

[1] RFC 1122, §1.1.3. "The transport layer provides end-to-end communication services for applications."
[2] RFC 1122, Requirements for Internet Hosts -- Communication Layers, IETF, R. Braden (Editor), October 1989
[3] RFC 3235, Network Address Translator (NAT)-Friendly Application Design Guidelines. D. Senie. January 2002.
[4] "ITU-T Recommendation X.224 (11/1995) ISO/IEC 8073" (http://www.itu.int/rec/T-REC-X.224-199511-I/en/). .

User Datagram Protocol

The **User Datagram Protocol** (**UDP**) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768.

UDP uses a simple transmission model without implicit handshaking dialogues for providing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.[1] If error correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

UDP's stateless nature is also useful for servers answering small queries from huge numbers of clients. Unlike TCP, UDP supports packet broadcast (sending to all on local network) and multicasting (send to all subscribers).[2]

Common network applications that use UDP include: the Domain Name System (DNS), streaming media applications such as IPTV, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP), IP tunneling protocols and many online games.

Service ports

UDP applications use datagram sockets to establish host-to-host communications. An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integer value, allowing for port numbers between 0 and 65535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

The Internet Assigned Numbers Authority has divided port numbers into three ranges.[2] Port numbers 0 through 1023 are used for common, well-known services. On Unix-like operating systems, using one of these ports requires superuser operating permission. Port numbers 1024 through 49151 are the registered ports used for IANA-registered services. Ports 49152 through 65535 are dynamic ports that are not officially designated for any specific service, and can be used for any purpose. They are also used as ephemeral ports, from which software running on the host may randomly choose a port in order to define itself.[2] In effect, they are used as temporary ports primarily by clients when communicating with servers.

Packet structure

UDP is a minimal message-oriented Transport Layer protocol that is documented in IETF RFC 768.

UDP provides no guarantees to the upper layer protocol for message delivery and the UDP protocol layer retains no state of UDP messages once sent. For this reason, UDP is sometimes referred to as *Unreliable* Datagram Protocol.[3]

UDP provides application multiplexing (via port numbers) and integrity verification (via checksum) of the header and payload.[4] If transmission reliability is desired, it must be implemented in the user's application.

offset (bits)	0 – 15	16 – 31
0	Source Port Number	Destination Port Number
32	Length	Checksum
64+	Data	

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits).[1] The use of two of those is optional in IPv4 (pink background in table). In IPv6 only the source port is optional (see below).

Source port number

This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.[2]

Destination port number

This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.[2]

Length

A field that specifies the length in bytes of the entire datagram: header and data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header).[2]

Checksum

The checksum field is used for error-checking of the header *and* data. If no checksum is generated by the transmitter, the field uses the value all-zeros.[5] This field is not optional for IPv6.[6]

Checksum computation

The method used to compute the checksum is defined in RFC 768:

> *Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.*[5]

In other words, all 16-bit words are summed using one's complement arithmetic. The sum is then one's complemented to yield the value of the UDP checksum field.

If the checksum calculation results in the value zero (all 16 bits 0) it should be sent as the one's complement (all 1s).

The difference between IPv4 and IPv6 is in the data used to compute the checksum.

IPv4 PSEUDO-HEADER

When UDP runs over IPv4, the checksum is computed using a PSEUDO-HEADER that contains some of the same information from the real IPv4 header. The PSEUDO-HEADER is not the real IPv4 header used to send an IP packet. The following table defines the PSEUDO-HEADER used only for the checksum calculation.

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Source address			
32	Destination address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160+	Data			

The source and destination addresses are those in the IPv4 header. The protocol is that for UDP (see *List of IP protocol numbers*): 17 (0x11). The UDP length field is the length of the UDP header and data.

UDP checksum computation is optional for IPv4. If a checksum is not used it should be set to the value zero.

IPv6 PSEUDO-HEADER

When UDP runs over IPv6, the checksum is mandatory. The method used to compute it is changed as documented in RFC 2460:

> *Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6 to include the 128-bit IPv6 addresses.*[6]

When computing the checksum, again a PSEUDO-HEADER is used that mimics the real IPv6 header:

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Source address			
32				
64				
96				
128	Destination address			
160				
192				
224				
256	UDP length			
288	Zeros			Next Header
320	Source Port		Destination Port	
352	Length		Checksum	
384+	Data			

The source address is the one in the IPv6 header. The destination address is the final destination; if the IPv6 packet does not contain a Routing header, that will be the destination address in the IPv6 header; otherwise, at the originating node, it will be the address in the last element of the Routing header, and, at the receiving node, it will be the destination address in the IPv6 header. The value of the Next Header field is the protocol value for UDP: 17. The UDP length field is the length of the UDP header and data.

Reliability and congestion control solutions

Lacking reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Some applications such as TFTP may add rudimentary reliability mechanisms into the application layer as needed.[2]

Most often, UDP applications do not employ reliability mechanisms and may even be hindered by them. Streaming media, real-time multiplayer games and voice over IP (VoIP) are examples of applications that often use UDP. In these particular applications, loss of packets is not usually a fatal problem. If an application requires a high degree of reliability, a protocol such as the Transmission Control Protocol or erasure codes may be used instead.

Potentially more seriously, unlike TCP, UDP-based applications don't necessarily have good congestion avoidance and control mechanisms. Congestion insensitive UDP applications that consume a large fraction of available bandwidth could endanger the stability of the internet, as they frequently give a bandwidth load that is inelastic. Network-based mechanisms have been proposed to minimize potential congestion collapse effects of uncontrolled, high rate UDP traffic loads. Network-based elements such as routers using packet queuing and dropping techniques are often the only tool available to slow down excessive UDP traffic. The Datagram Congestion Control Protocol (DCCP) is being designed as a partial solution to this potential problem by adding end host TCP-friendly congestion control behavior to high-rate UDP streams such as streaming media.

Applications

Numerous key Internet applications use UDP, including: the Domain Name System (DNS), where queries must be fast and only consist of a single request followed by a single reply packet, the Simple Network Management Protocol (SNMP), the Routing Information Protocol (RIP)[1] and the Dynamic Host Configuration Protocol (DHCP).

Voice and video traffic is generally transmitted using UDP. Real-time video and audio streaming protocols are designed to handle occasional lost packets, so only slight degradation in quality occurs, rather than large delays if lost packets were retransmitted. Because both TCP and UDP run over the same network, many businesses are finding that a recent increase in UDP traffic from these real-time applications is hindering the performance of applications using TCP, such as point of sale, accounting, and database systems. When TCP detects packet loss, it will throttle back its data rate usage. Since both real-time and business applications are important to businesses, developing quality of service solutions is seen as crucial by some.[7]

Comparison of UDP and TCP

Transmission Control Protocol is a connection-oriented protocol, which means that it requires handshaking to set up end-to-end communications. Once a connection is set up user data may be sent bi-directionally over the connection.

- *Reliable* – TCP manages message acknowledgment, retransmission and timeout. Multiple attempts to deliver the message are made. If it gets lost along the way, the server will re-request the lost part. In TCP, there's either no missing data, or, in case of multiple timeouts, the connection is dropped.
- *Ordered* – if two messages are sent over a connection in sequence, the first message will reach the receiving application first. When data segments arrive in the wrong order, TCP buffers the out-of-order data until all data can be properly re-ordered and delivered to the application.
- *Heavyweight* – TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.
- *Streaming* – Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.

UDP is a simpler message-based connectionless protocol. Connectionless protocols do not set up a dedicated end-to-end connection. Communication is achieved by transmitting information in one direction from source to destination without verifying the readiness or state of the receiver. However, one primary benefit of UDP over TCP is the application to voice over internet protocol (VoIP) where any handshaking would hinder clear voice

communication. It is assumed in VoIP UDP that the end users provide any necessary real time confirmation that the message has been received.

- *Unreliable* – When a message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission or timeout.
- *Not ordered* – If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.
- *Lightweight* – There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
- *Datagrams* – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
- *No congestion control* – UDP itself does not avoid congestion, and it's possible for high bandwidth applications to trigger congestion collapse, unless they implement congestion control measures at the application level.

References

[1] Kurose, J. F.; Ross, K. W. (2010). *Computer Networking: A Top-Down Approach* (5th ed.). Boston, MA: Pearson Education. ISBN 9780131365483.

[2] Forouzan, B.A. (2000). *TCP/IP: Protocol Suite, 1st ed*. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.

[3] content@ipv6.com. "UDP Protocol Overview" (http://ipv6.com/articles/general/User-Datagram-Protocol.htm). Ipv6.com. . Retrieved 2011-08-17.

[4] Clark, M.P. (2003). *Data Networks IP and the Internet, 1st ed*. West Sussex, England: John Wiley & Sons Ltd.

[5] Postel, J. (August 1980). RFC 768: User Datagram Protocol. *Internet Engineering Task Force*. Retrieved from http://tools.ietf.org/html/rfc768

[6] Deering S. & Hinden R. (December 1998). RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. *Internet Engineering Task Force*. Retrieved from http://tools.ietf.org/html/rfc2460

[7] "The impact of UDP on Data Applications" (http://www.networkperformancedaily.com/2007/08/whiteboard_series_nice_guys_fi.html). Networkperformancedaily.com. . Retrieved 2011-08-17.

RFC references

- RFC 768 – User Datagram Protocol
- RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification
- RFC 4113 – Management Information Base for the UDP
- RFC 5405 – Unicast UDP Usage Guidelines for Application Designers

External links

- IANA Port Assignments (http://www.iana.org/assignments/port-numbers)
- The Trouble with UDP Scanning (PDF) (http://condor.depaul.edu/~jkristof/papers/udpscanning.pdf)
- Breakdown of UDP frame (http://www.networksorcery.com/enp/protocol/udp.htm)
- UDP on MSDN Magazine Sockets and WCF (http://msdn.microsoft.com/en-us/magazine/cc163648.aspx)
- UDP connections (http://www.faqs.org/docs/iptables/udpconnections.html)

Routing Protocols in Different Layers

Adaptive routing

Adaptive routing describes the capability of a system, through which routes are characterized by their destination, to alter the path that the route takes through the system in response to a change in conditions[1] . The adaptation is intended to allow as many routes as possible to remain valid (that is, have destinations that can be reached) in response to the change.

People using a transport system can display adaptive routing. For example, if a local railway station is closed, people can alight from a train at a different station and use another method, such as a bus, to reach their destination. Another example of adaptive routing can be seen within financial markets. For example, ASOR or Adaptive Smart Order Router (developed by Quod Financial), takes routing decisions dynamically and based on real-time market events.

The term is commonly used in data networking to describe the capability of a network to 'route around' damage, such as loss of a node or a connection between nodes, so long as other path choices are available. There are several protocols used to achieve this:

- RIP
- OSPF
- IS-IS
- IGRP/EIGRP

Systems that do not implement adaptive routing are described as using static routing, where routes through a network are described by fixed paths (statically). A change, such as the loss of a node, or loss of a connection between nodes, is not compensated for. This means that anything that wishes to take an affected path will either have to wait for the failure to be repaired before restarting its journey, or will have to fail to reach its destination and give up the journey.

References

[1] http://www-net.cs.yale.edu/publications/XQYZ04.pdf

Bellman–Ford algorithm

Bellman–Ford algorithm

Class	Single-source shortest path problem (for weighted directed graphs)
Data structure	Graph
Worst case performance	$O(\|V\|\|E\|)$
Worst case space complexity	$O(\|V\|)$

The **Bellman–Ford algorithm** computes single-source shortest paths in a weighted digraph. For graphs with only non-negative edge weights, the faster Dijkstra's algorithm also solves the problem. Thus, Bellman–Ford is used primarily for graphs with negative edge weights. The algorithm is named after its developers, Richard Bellman and Lester Ford, Jr.

If a graph contains a "negative cycle", i.e., a cycle whose edges sum to a negative value, then walks of arbitrarily low weight can be constructed, i.e., there may be no *shortest* path. Bellman-Ford can detect negative cycles and report their existence, but it cannot produce a correct answer if a negative cycle is reachable from the source.

According to Robert Sedgewick, "Negative weights are not merely a mathematical curiosity; [...] [they] arise in a natural way when we reduce other problems to shortest-paths problems".[1] Let G be a graph containing a negative cycle. One NP-Complete variant of the shortest-path problem asks for the shortest path in G (containing a negative cycle) such that no edge is repeated. Sedgewick gives a reduction from the Hamiltonian path problem to this variant of the problem.

Algorithm

Bellman–Ford is in its basic structure very similar to Dijkstra's algorithm, but instead of greedily selecting the minimum-weight node not yet processed to relax, it simply relaxes *all* the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph. The repetitions allow minimum distances to accurately propagate throughout the graph, since, in the absence of negative cycles, the shortest path can only visit each node at most once. Unlike the greedy approach, which depends on certain structural assumptions derived from positive weights, this straightforward approach extends to the general case.

Bellman–Ford runs in $O(|V|\cdot|E|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges respectively.

```
procedure BellmanFord(list vertices, list edges, vertex source)
   // This implementation takes in a graph, represented as lists of vertices
   // and edges, and modifies the vertices so that their distance and
   // predecessor attributes store the shortest paths.

   // Step 1: initialize graph
   for each vertex v in vertices:
       if v is source then v.distance := 0
       else v.distance := infinity
       v.predecessor := null

   // Step 2: relax edges repeatedly
   for i from 1 to size(vertices)-1:
       for each edge uv in edges: // uv is the edge from u to v
```

```
        u := uv.source
        v := uv.destination
        if u.distance + uv.weight < v.distance:
            v.distance := u.distance + uv.weight
            v.predecessor := u

// Step 3: check for negative-weight cycles
for each edge uv in edges:
    u := uv.source
    v := uv.destination
    if u.distance + uv.weight < v.distance:
        error "Graph contains a negative-weight cycle"
```

Proof of correctness

The correctness of the algorithm can be shown by induction. The precise statement shown by induction is:

Lemma. After *i* repetitions of *for* cycle:

- If Distance(*u*) is not infinity, it is equal to the length of some path from *s* to *u*;
- If there is a path from *s* to *u* with at most *i* edges, then Distance(u) is at most the length of the shortest path from *s* to *u* with at most *i* edges.

Proof. For the base case of induction, consider `i=0` and the moment before *for* cycle is executed for the first time. Then, for the source vertex, `source.distance = 0`, which is correct. For other vertices *u*, `u.distance = infinity`, which is also correct because there is no path from *source* to *u* with 0 edges.

For the inductive case, we first prove the first part. Consider a moment when a vertex's distance is updated by `v.distance := u.distance + uv.weight`. By inductive assumption, `u.distance` is the length of some path from *source* to *u*. Then `u.distance + uv.weight` is the length of the path from *source* to *v* that follows the path from *source* to *u* and then goes to *v*.

For the second part, consider the shortest path from *source* to *u* with at most *i* edges. Let *v* be the last vertex before *u* on this path. Then, the part of the path from *source* to *v* is the shortest path from *source* to *v* with at most *i-1* edges. By inductive assumption, `v.distance` after $i-1$ cycles is at most the length of this path. Therefore, `uv.weight + v.distance` is at most the length of the path from *s* to *u*. In the ith cycle, `u.distance` gets compared with `uv.weight + v.distance`, and is set equal to it if `uv.weight + v.distance` was smaller. Therefore, after *i* cycles, `u.distance` is at most the length of the shortest path from *source* to *u* that uses at most *i* edges.

If there are no negative-weight cycles, then every shortest path visits each vertex at most once, so at step 3 no further improvements can be made. Conversely, suppose no improvement can be made. Then for any cycle with vertices $v[0], ..., v[k-1]$,

```
v[i].distance <= v[(i-1) mod k].distance + v[(i-1) mod k]v[i].weight
```

Summing around the cycle, the $v[i]$.distance terms and the $v[i-1 \pmod k]$ distance terms cancel, leaving

```
0 <= sum from 1 to k of v[i-1 (mod k)]v[i].weight
```

I.e., every cycle has nonnegative weight.

Applications in routing

A distributed variant of the Bellman–Ford algorithm is used in distance-vector routing protocols, for example the Routing Information Protocol (RIP). The algorithm is distributed because it involves a number of nodes (routers) within an Autonomous system, a collection of IP networks typically owned by an ISP. It consists of the following steps:

1. Each node calculates the distances between itself and all other nodes within the AS and stores this information as a table.
2. Each node sends its table to all neighboring nodes.
3. When a node receives distance tables from its neighbors, it calculates the shortest routes to all other nodes and updates its own table to reflect any changes.

The main disadvantages of the Bellman–Ford algorithm in this setting are as follows:

- It does not scale well.
- Changes in network topology are not reflected quickly since updates are spread node-by-node.
- Count to infinity (if link or node failures render a node unreachable from some set of other nodes, those nodes may spend forever gradually increasing their estimates of the distance to it, and in the meantime there may be routing loops).

Yen's improvement

Yen (1970) described an improvement to the Bellman–Ford algorithm for a graph without negative-weight cycles. Yen's improvement first assigns some arbitrary linear order on all vertices and then partitions the set of all edges into one of two subsets. The first subset, E_f, contains all edges (v_i, v_j) such that $i < j$; the second, E_b, contains edges (v_i, v_j) such that $i > j$. Each vertex is visited in the order $v_1, v_2, ..., v_{|V|}$, relaxing each outgoing edge from that vertex in E_f. Each vertex is then visited in the order $v_{|V|}, v_{|V|-1}, ..., v_1$, relaxing each outgoing edge from that vertex in E_b. Yen's improvement effectively halves the number of "passes" required for the single-source-shortest-path solution.

Notes

[1] Robert Sedgewick. Algorithms in Java. Third Edition. ISBN 0-201-36121-3. Section 21.7: Negative Edge Weights. http://safari.oreilly.com/0201361213/ch21lev1sec7

References

- Bellman, Richard (1958), "On a routing problem", *Quarterly of Applied Mathematics* **16**: 87–90, MR0102435.
- Ford, L. R., Jr.; Fulkerson, D. R. (1962), *Flows in Networks*, Princeton University Press.
- Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L.. *Introduction to Algorithms*. MIT Press and McGraw-Hill., Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.1: The Bellman–Ford algorithm, pp. 588–592. Problem 24-1, pp. 614–615.
- Heineman, George T.; Pollice, Gary; Selkow, Stanley (2008), "Chapter 6: Graph Algorithms", *Algorithms in a Nutshell*, O'Reilly Media, pp. 160–164, ISBN 978-0-596-51624-6
- Yen, Jin Y. (1970), "An algorithm for finding shortest routes from all source nodes to a given destination in general networks", *Quarterly of Applied Mathematics* **27**: 526–530, MR0253822.

External links

- C code example (http://snippets.dzone.com/posts/show/4828)
- Open Source Java Graph package with Bellman-Ford Algorithms (http://code.google.com/p/annas/)

Border Gateway Protocol

The **Border Gateway Protocol** (**BGP**) is the protocol backing the core routing decisions on the Internet. It maintains a table of IP networks or 'prefixes' which designate network reachability among autonomous systems (AS). It is described as a path vector protocol. BGP does not use traditional Interior Gateway Protocol (**IGP**) metrics, but makes routing decisions based on path, network policies and/or rulesets. For this reason, it is more appropriately termed a reachability protocol rather than routing protocol.

BGP was created to replace the Exterior Gateway Protocol (**EGP**) protocol to allow fully decentralized routing in order to transition from the core ARPAnet model to a decentralized system that included the NSFNET backbone and its associated regional networks. This allowed the Internet to become a truly decentralized system. Since 1994, version four of the BGP has been in use on the Internet. All previous versions are now obsolete. The major enhancement in version 4 was support of Classless Inter-Domain Routing and use of route aggregation to decrease the size of routing tables. Since January 2006, version 4 is codified in RFC 4271, which went through more than 20 drafts based on the earlier RFC 1771 version 4. RFC 4271 version corrected a number of errors, clarified ambiguities and brought the RFC much closer to industry practices.

Most Internet service providers must use BGP to establish routing between one another (especially if they are multihomed). Therefore, even though most Internet users do not use it directly, BGP is one of the most important protocols of the Internet. Compare this with Signaling System 7 (SS7), which is the inter-provider core call setup protocol on the PSTN. Very large private IP networks use BGP internally. An example would be the joining of a number of large OSPF (Open Shortest Path First) networks where OSPF by itself would not scale to size. Another reason to use BGP is multihoming a network for better redundancy either to multiple access points of a single ISP (RFC 1998) or to multiple ISPs.

Operation

BGP neighbors, called peers, are established by manual configuration between routers to create a TCP session on port 179. A BGP speaker will periodically send 19-byte keep-alive messages to maintain the connection (every 60 seconds by default). Among routing protocols, BGP is unique in using TCP as its transport protocol.

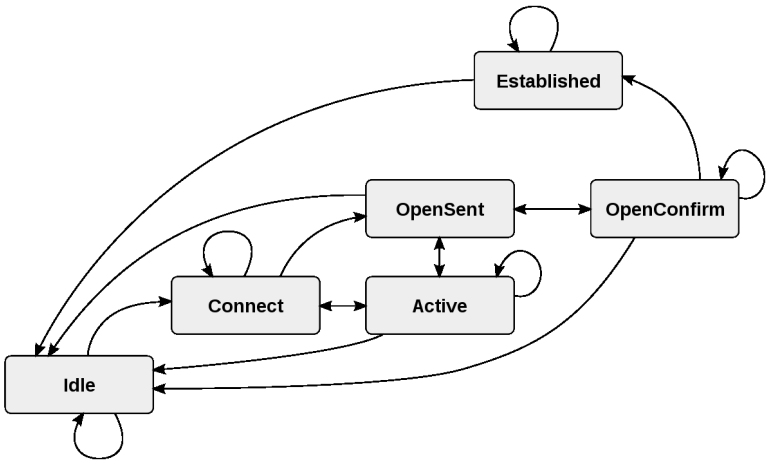
When BGP runs between two peers in the same autonomous system (AS), it is referred to as *Internal BGP* (*IBGP* or *Interior Border Gateway Protocol*). When it runs between autonomous systems, it is called *External BGP* (*EBGP* or *Exterior Border Gateway Protocol*). Routers on the boundary of one AS exchanging information with another AS are called border or edge routers. In the Cisco operating system, IBGP routes have an administrative distance of 200, which is less preferred than either external BGP or any interior routing protocol. Other router implementations also prefer EBGP to IGPs, and IGPs to IBGP.

Extensions negotiation

During the OPEN, BGP speakers can negotiate[1] optional capabilities of the session, including multiprotocol extensions and various recovery modes. If the multiprotocol extensions to BGP[2] are negotiated at the time of creation, the BGP speaker can prefix the Network Layer Reachability Information (NLRI) it advertises with an address family prefix. These families include the IPv4 (default), IPv6, IPv4/IPv6 Virtual Private Networks and multicast BGP. Increasingly, BGP is used as a generalized signaling protocol to carry information about routes that

may not be part of the global Internet, such as VPNs.[3]

Finite-state machine



BGP state machine

In order to make decisions in its operations with other BGP peers, a BGP peer uses a simple finite state machine (FSM) that consists of six states: Idle; Connect; Active; OpenSent; OpenConfirm; and Established. For each peer-to-peer session, a BGP implementation maintains a state variable that tracks which of these six states the session is in. The BGP protocol defines the messages that each peer should exchange in order to change the session from one state to another. The first state is the “Idle” state. In the “Idle” state, BGP initializes all resources, refuses all inbound BGP connection attempts and initiates a TCP connection to the peer. The second state is “Connect”. In the “Connect” state, the router waits for the TCP connection to complete and transitions to the "OpenSent" state if successful. If unsuccessful, it starts the ConnectRetry timer and transitions to the "Active" state upon expiration. In the "Active" state, the router resets the ConnectRetry timer to zero and returns to the "Connect" state. In the "OpenSent" state, the router sends an Open message and waits for one in return in order to transition to the "OpenConfirm" state. Keepalive messages are exchanged and, upon successful receipt, the router is placed into the “Established” state. In the “Established” state, the router can send/receive: Keepalive; Update; and Notification messages to/from its peer.

- **Idle State**:
  - Refuse all incoming BGP connections
  - Start event triggers the initialization of
  - Initiates a TCP connection with its configured BGP peer.
  - Listens for a TCP connection from its peer.
  - Changes its state to Connect.
  - If an error occurs at any state of the FSM process, the BGP session is terminated immediately and returned to the Idle state. Some of the reasons why a router does not progress from the Idle state are:
    - TCP port 179 is not open.
    - A random TCP port over 1023 is not open.
    - Peer address configured incorrectly on either router.
    - AS number configured incorrectly on either router .
- **Connect State**:
  - Waits for successful TCP negotiation with peer.
  - BGP does not spend much time in this state if the TCP session has been successfully established.
  - Sends Open message to peer and changes state to OpenSent.
  - If an error occurs, BGP moves to the Active state. Some reasons for the error are:
    - TCP port 179 is not open.
    - A random TCP port over 1023 is not open.
    - Peer address configured incorrectly on either router.

    - AS number configured incorrectly on either router.
- **Active State**:
  - If the router was unable to establish a successful TCP session, then it ends up in the Active state.
  - BGP FSM will try to restart another TCP session with the peer and, if successful, then it will send an Open message to the peer.
  - If it is unsuccessful again, the FSM is reset to the Idle state.
  - Repeated failures may result in a router cycling between the Idle and Active states. Some of the reasons for this include:
    - TCP port 179 is not open.
    - A random TCP port over 1023 is not open.
    - BGP configuration error.
    - Network congestion.
    - Flapping network interface.
- **OpenSent State**:
  - BGP FSM listens for an Open message from its peer.
  - Once the message has been received, the router checks the validity of the Open message.
  - If there is an error it is because one of the fields in the Open message doesn't match between the peers, e.g. BGP version mismatch, MD5 password mismatch, the peering router expects a different My AS. The router will then send a Notification message to the peer indicating why the error occurred.
  - If there is no error, a Keepalive message is sent, various timers are set and the state is changed to OpenConfirm.
- **OpenConfirm State**:
  - The peer is listening for a Keepalive message from its peer.
  - If a Keepalive message is received and no timer has expired before reception of the Keepalive, BGP transitions to the Established state.
  - If a timer expires before a Keepalive message is received, or if an error condition occurs, the router transitions back to the Idle state.
- **Established State**:
  - In this state, the peers send Update messages to exchange information about each route being advertised to the BGP peer.
  - If there is any error in the Update message then a Notification message is sent to the peer, and BGP transitions back to the Idle state.
  - If a timer expires before a Keepalive message is received, or if an error condition occurs, the router transitions back to the Idle state.

BGP router connectivity and learning routes

In the simplest arrangement all routers within a single AS and participating in BGP routing must be configured in a full mesh: each router must be configured as peer to every other router. This causes scaling problems, since the number of required connections grows quadratically with the number of routers involved. To alleviate the problem, BGP implements two options: route reflectors (RFC 4456) and confederations (RFC 5065). The following discussion of basic UPDATE processing assumes a full IBGP mesh.

Basic update processing

A given BGP router may accept NLRI in UPDATEs from multiple neighbors and advertise NLRI to the same, or a different set, of neighbors. Conceptually, BGP maintains its own "master" routing table, called the *Loc-RIB* (Local Routing Information Base), separate from the main routing table of the router. For each neighbor, the BGP process maintains a conceptual *Adj-RIB-In* (Adjacent Routing Information Base, Incoming) containing the NLRI received from the neighbor, and a conceptual *Adj-RIB-Out* (Outgoing) for NLRI to be sent to the neighbor.

Conceptual, in the preceding paragraph, means that the physical storage and structure of these various tables are decided by the implementer of the BGP code. Their structure is not visible to other BGP routers, although they usually can be interrogated with management commands on the local router. It is quite common, for example, to store the two Adj-RIBs and the Loc-RIB together in the same data structure, with additional information attached to the RIB entries. The additional information tells the BGP process such things as whether individual entries belong in the Adj-RIBs for specific neighbors, whether the per-neighbor route selection process made received policies eligible for the Loc-RIB, and whether Loc-RIB entries are eligible to be submitted to the local router's routing table management process.

By *eligible to be submitted*, BGP will submit the routes that it considers best to the main routing table process. Depending on the implementation of that process, the BGP route is not necessarily selected. For example, a directly connected prefix, learned from the router's own hardware, is usually most preferred. As long as that directly connected route's interface is active, the BGP route to the destination will not be put into the routing table. Once the interface goes down, and there are no more preferred routes, the Loc-RIB route would be installed in the main routing table. Until recently, it was a common mistake to say *BGP carries policies*. BGP actually carried the information with which rules inside BGP-speaking routers could make policy decisions. Some of the information carried that is explicitly intended to be used in policy decisions are communities and multi-exit discriminators (MED).

Route selection

The BGP standard specifies a number of decision factors, more than are used by any other common routing process, for selecting NLRI (Network Layer Reachability Information) to go into the Loc-RIB (Routing Information Base). The first decision point for evaluating NLRI is that its next-hop attribute must be reachable (or resolvable). Another way of saying the next-hop must be reachable is that there must be an active route, already in the main routing table of the router, to the prefix in which the next-hop address is located.

Next, for each neighbor, the BGP process applies various standard and implementation-dependent criteria to decide which routes conceptually should go into the Adj-RIB-In. The neighbor could send several possible routes to a destination, but the first level of preference is at the neighbor level. Only one route to each destination will be installed in the conceptual Adj-RIB-In. This process will also delete, from the Adj-RIB-In, any routes that are withdrawn by the neighbor.

Whenever a conceptual Adj-RIB-In changes, the main BGP process decides if any of the neighbor's new routes are preferred to routes already in the Loc-RIB. If so, it replaces them. If a given route is withdrawn by a neighbor, and there is no other route to that destination, the route is removed from the Loc-RIB, and no longer sent, by BGP, to the main routing table manager. If the router does not have a route to that destination from any non-BGP source, the withdrawn route will be removed from the main routing table.

Per-neighbor decisions

After verifying that the next hop is reachable, if the route comes from an internal (i.e. IBGP) peer, the first rule to apply according to the standard is to examine the LOCAL_PREF attribute. If there are several IBGP routes from the neighbor, the one with the highest LOCAL_PREF is selected unless there are several routes with the same LOCAL_PREF. In the latter case the route selection process moves to the next tie breaker. While LOCAL_PREF is the first rule in the standard, once reachability of the NEXT_HOP is verified, Cisco and several other vendors first consider a decision factor called WEIGHT which is local to the router (i.e. not transmitted by BGP). The route with the highest WEIGHT is preferred.

LOCAL_PREF, WEIGHT, and other criteria can be manipulated by local configuration and software capabilities. Such manipulation is outside the scope of the standard but is commonly used. For example the COMMUNITY attribute (see below) is not directly used by the BGP selection process. The BGP neighbor process however can have a rule to set LOCAL_PREFERENCE or another factor based on a manually programmed rule to set the attribute if the COMMUNITY value matches some pattern matching criterion. If the route was learned from an external peer the per-neighbor BGP process computes a LOCAL_PREFERENCE value from local policy rules and then compares the LOCAL_PREFERENCE of all routes from the neighbor.

At the per-neighbor level - ignoring implementation-specific policy modifiers - the order of tie breaking rules is:

1. Prefer the route with the shortest AS_PATH. An AS_PATH is the set of AS numbers that must be traversed to reach the advertised destination. AS1-AS2-AS3 is shorter than AS4-AS5-AS6-AS7.
2. Prefer routes with the lowest value of their ORIGIN attribute.
3. Prefer routes with the lowest MULTI_EXIT_DISC (multi-exit discriminator or MED) value.

Before the most recent edition of the BGP standard, if an UPDATE had no MULTI_EXIT_DISC value, several implementations created a MED with the least possible value. The current standard however specifies that missing MEDs are to be treated as the highest possible value. Since the current rule may cause different behavior than the vendor interpretations, BGP implementations that used the nonstandard default value have a configuration feature that allows the old or standard rule to be selected.

Decision factors at the Loc-RIB level

Once candidate routes are received from neighbors, the Loc-RIB software applies additional tie-breakers to routes to the same destination.

1. If at least one route was learned from an external neighbor (i.e., the route was learned from EBGP), drop all routes learned from IBGP.
2. Prefer the route with the lowest interior cost to the NEXT_HOP, according to the main Routing Table. If two neighbors advertised the same route, but one neighbor is reachable via a low-bitrate link and the other by a high-bitrate link, and the interior routing protocol calculates lowest cost based on highest bitrate, the route through the high-bitrate link would be preferred and other routes dropped.

If there is more than one route still tied at this point, several BGP implementations offer a configurable option to load-share among the routes, accepting all (or all up to some number).

1. Prefer the route learned from the BGP speaker with the numerically lowest BGP identifier
2. Prefer the route learned from the BGP speaker with the lowest peer IP address

Communities

BGP communities are attribute tags that can be applied to incoming or outgoing prefixes to achieve some common goal (RFC 1997). While it is common to say that BGP allows an administrator to set policies on how prefixes are handled by ISPs, this is generally not possible, strictly speaking. For instance, BGP natively has no concept to allow one AS to tell another AS to restrict advertisement of a prefix to only North American peering customers. Instead, an ISP generally publishes a list of well-known or proprietary communities with a description for each one, which essentially becomes an agreement of how prefixes are to be treated. Examples of common communities include local preference adjustments, geographic or peer type restrictions, DoS avoidance (black holing), and AS prepending options. An ISP might state that any routes received from customers with community XXX:500 will be advertised to all peers (default) while community XXX:501 will restrict advertisement to North America. The customer simply adjusts their configuration to include the correct community(ies) for each route, and the ISP is responsible for controlling who the prefix is advertised to. The end user has no technical ability to enforce correct actions being taken by the ISP, though problems in this area are generally rare and accidental.

It is a common tactic for end customers to use BGP communities (usually ASN:70,80,90,100) to control the local preference the ISP assigns to advertised routes instead of using MED (the effect is similar). It should also be noted that the community attribute is transitive, but communities applied by the customer very rarely become propagated outside the next-hop AS.

Extended communities

The BGP Extended Community Attribute was added in 2006 in order to extend the range of such attributes and to provide a community attribute structuring by means of a type field. The extended format consists of one or two octets for the type field followed by seven or six octets for the respective community attribute content. The definition of this Extended Community Attribute is documented in RFC 4360. The IANA administers the registry for BGP Extended Communities Types.[4] The Extended Communities Attribute itself is a transitive optional BGP attribute. However, a bit in the type field within the attribute decides whether the encoded extended community is of a transitive or non-transitive nature. The IANA registry therefore provides different number ranges for the attribute types. Due to the extended attribute range, its usage can be manifold. RFC 4360 exemplarly defines the "Two-Octet AS Specific Extended Community", the "IPv4 Address Specific Extended Community", the "Opaque Extended Community", the "Route Target Community" and the "Route Origin Community". A number of BGP QoS drafts[5] also use this Extended Community Attribute structure for inter-domain QoS signalling.

Uses of multi-exit discriminators

MEDs, defined in the main BGP standard, were originally intended to show to another neighbor AS the advertising AS's preference as to which of several links are preferred for inbound traffic. Another application of MEDs is to advertise the value, typically based on delay, of multiple AS that have presence at an IXP, that they impose to send traffic to some destination.

Message header format

The following is the BGP version 4 message header format:

bit offset	0–15	16–23	24–31
0	Marker		
32			
64			
96			
128	Length	Type	

- **Marker**: Included for compatibility, must be set to all ones.
- **Length**: Total length of the message in octets, including the header.
- **Type**: Type of BGP message. The following values are defined:
  - Open (1)
  - Update (2)
  - Notification (3)
  - KeepAlive (4)

BGP problems and mitigation

Internal BGP scalability

An autonomous system with internal BGP (IBGP) must have all of its IBGP peers connect to each other in a full mesh (where everyone speaks to everyone directly). This full-mesh configuration requires that each router maintain a session to every other router. In large networks, this number of sessions may degrade performance of routers, due either to a lack of memory, or too much CPU process requirements.

Route reflectors and confederations both reduce the number of IBGP peers to each router and thus reduce processing overhead. Route reflectors are a pure performance-enhancing technique, while confederations also can be used to implement more fine-grained policy.

Route reflectors[6] reduce the number of connections required in an AS. A single router (or two for redundancy) can be made a route reflector: other routers in the AS need only be configured as peers to them.

Confederations are sets of autonomous systems. In common practice,[7] only one of the confederation AS numbers is seen by the Internet as a whole. Confederations are used in very large networks where a large AS can be configured to encompass smaller more manageable internal ASs.

Confederations can be used in conjunction with route reflectors. Both confederations and route reflectors can be subject to persistent oscillation unless specific design rules, affecting both BGP and the interior routing protocol, are followed.[8]

However, these alternatives can introduce problems of their own, including the following:

- route oscillation
- sub-optimal routing
- increase of BGP convergence time[9]

Additionally, route reflectors and BGP confederations were not designed to ease BGP router configuration. Nevertheless, these are common tools for experienced BGP network architects. These tools may be combined, for example, as a hierarchy of route reflectors.

Instability

The routing tables managed by a BGP implementation are adjusted continually to reflect actual changes in the network, such as links breaking and being restored or routers going down and coming back up. In the network as a whole it is normal for these changes to happen almost continuously, but for any particular router or link changes are supposed to be relatively infrequent. If a router is misconfigured or mismanaged then it may get into a rapid cycle between down and up states. This pattern of repeated withdrawal and re-announcement known as route flapping can cause excessive activity in all the other routers that know about the broken link, as the same route is continuously injected and withdrawn from the routing tables. The BGP design is such that delivery of traffic may not function while routes are being updated. On the Internet, a BGP routing change may cause outages for several minutes.

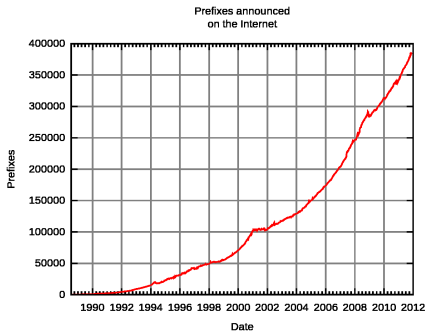
A feature known as *route flap damping* (RFC 2439 [10]) is built into many BGP implementations in an attempt to mitigate the effects of route flapping. Without damping the excessive activity can cause a heavy processing load on routers, which may in turn delay updates on other routes, and so affect overall routing stability. With damping, a route's flapping is exponentially decayed. At the first instance when a route becomes unavailable and quickly reappears, damping does not take effect, so as to maintain the normal fail-over times of BGP. At the second occurrence, BGP shuns that prefix for a certain length of time; subsequent occurrences are timed out exponentially. After the abnormalities have ceased and a suitable length of time has passed for the offending route, prefixes can be reinstated and its slate wiped clean. Damping can also mitigate denial of service attacks; damping timings are highly customizable.

It is also suggested in RFC 2439 (under "Design Choices -> Stability Sensitive Suppression of Route Advertisement") that route flap dampening is a feature more desirable if implemented to Exterior Border Gateway Protocol Sessions (EBGP sessions or simply called exterior peers) and not on Interior Border Gateway Protocol Sessions (IBGP sessions or simply called internal peers); With this approach when a route flaps inside an autonomous system, it is not propagated to the external ASs - flapping a route to an EBGP will have a chain of flapping for the particular route throughout the backbone. This method also successfully avoids the overhead of route flap dampening for IBGP sessions.

However, subsequent research has shown that flap damping can actually lengthen convergence times in some cases, and can cause interruptions in connectivity even when links are not flapping.[11] [12] Moreover, as backbone links and router processors have become faster, some network architects have suggested that flap damping may not be as important as it used to be, since changes to the routing table can be absorbed much faster by routers. This has led the RIPE Route Working Group to write that "with the current implementations of BGP flap damping, the application of flap damping in ISP networks is NOT recommended. ... If flap damping is implemented, the ISP operating that network will cause side-effects to their customers and the Internet users of their customers' content and services ... . These side-effects would quite likely be worse than the impact caused by simply not running flap damping at all."[13] Improving stability without the problems of flap damping is the subject of current research.[14]

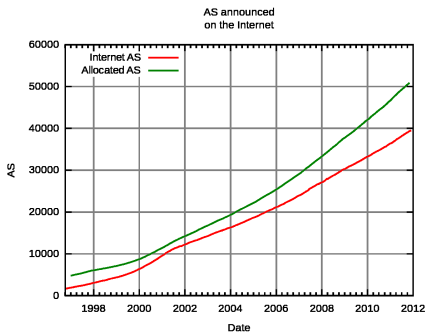
Routing table growth

One of the largest problems faced by BGP, and indeed the Internet infrastructure as a whole, is the growth of the Internet routing table. If the global routing table grows to the point where some older, less capable, routers cannot cope with the memory requirements or the CPU load of maintaining the table, these routers will cease to be effective gateways between the parts of the Internet they connect. In addition, and perhaps even more importantly, larger routing tables take longer to stabilize (see above) after a major connectivity change, leaving network service unreliable, or even unavailable, in the interim.



BGP table growth on the Internet.

Until late 2001, the global routing table was growing exponentially, threatening an eventual widespread breakdown of connectivity. In an attempt to prevent this, ISPs cooperated in keeping the global routing table as small as possible, by using Classless Inter-Domain Routing (CIDR) and route aggregation. While this slowed the growth of the routing table to a linear process for several years, with the expanded demand for multihoming by end user networks the growth was once again superlinear by the middle of 2004. As of April 2010, the routing table has in excess of 310,000 entries.[15]



Number of AS on the Internet vs number of registered AS.

Route summarization is often used to improve aggregation of the BGP global routing table, thereby reducing the necessary table size in routers of an AS. Consider AS1 has been allocated the big address space of 172.16.0.0/16, this would be counted as one route in the table, but due to customer requirement or traffic engineering purposes, AS1 wants to announce smaller, more specific routes of 172.16.0.0/18, 172.16.64.0/18 and 172.16.128.0/18. The prefix 172.16.192.0/18 does not have any hosts so AS1 does not announce a specific route 172.16.192.0/18. This all counts as AS1 announcing four routes.

AS2 will see the 4 routes from AS1 (172.16.0.0/16, 172.16.0.0/18, 172.16.64.0/18 and 172.16.128.0/18) and it is up to the routing policy of AS2 to decide whether or not to take a copy of the four routes or, as 172.16.0.0/16 overlaps all the other specific routes, to just store the summary, 172.16.0.0/16.

If AS2 wants to send data to prefix 172.16.192.0/18, it will be sent to the routers of AS1 on route 172.16.0.0/16. At AS1's router, it will either be dropped or a destination unreachable ICMP message will be sent back, depending on the configuration of AS1's routers.

If AS1 later decides to drop the route 172.16.0.0/16, leaving 172.16.0.0/18, 172.16.64.0/18 and 172.16.128.0/18, AS1 will drop the number of routes it announces to three. AS2 will see the three routes, and depending on the routing policy of AS2, it will store a copy of the three routes, or aggregate the prefix's 172.16.0.0/18 and 172.16.64.0/18 to 172.16.0.0/17, thereby reducing the number of routes AS2 stores to only two: 172.16.0.0/17 and 172.16.128.0/18.

If AS2 wants to send data to prefix 172.16.192.0/18, it will be dropped or a destination unreachable ICMP message will be sent back at the routers of AS2 (not AS1 as before), because 172.16.192.0/18 would not be in the routing table.

Load-balancing problem

Another factor causing this growth of the routing table is the need for load balancing of multi-homed networks. It is not a trivial task to balance the inbound traffic to a multi-homed network across its multiple inbound paths, due to limitation of the BGP route selection process. For a multi-homed network, if it announces the same network blocks across all of its BGP peers, the result may be that one or several of its inbound links become congested while the other links remain under-utilized, because external networks all picked that set of congested paths as optimal. Like most other routing protocols, the BGP protocol does not detect congestion.

To work around this problem, BGP administrators of that multihomed network may divide a large continuous IP address block into smaller blocks, and tweak the route announcement to make different blocks look optimal on different paths, so that external networks will choose a different path to reach different blocks of that multi-homed network. Such cases will increase the number of routes as seen on the global BGP table.

IP Hijacking

IP hijacking (sometimes referred to as BGP hijacking, prefix hijacking or route hijacking) is the illegitimate take over of groups of IP addresses by corrupting Internet routing tables. This can be caused both by programming errors, attacks on a network or attempts at censorship. This will cause internet sites to go out of service, as routers will no longer find them.

Requirements of a router for use of BGP for Internet and backbone-of-backbones purposes

Routers, especially small ones intended for Small Office/Home Office (SOHO) use, may not include BGP software. Some SOHO routers simply are not capable of running BGP using BGP routing tables of any size. Other commercial routers may need a specific software executable image that contains BGP, or a license that enables it. Open source packages that run BGP include GNU Zebra, Quagga, OpenBGPD, BIRD, XORP and Vyatta. Devices marketed as Layer 3 switches are less likely to support BGP than devices marketed as routers, but high-end Layer 3 Switches usually can run BGP.

Products marketed as switches may or may not have a size limitation on BGP tables, such as 20,000 routes, far smaller than a full Internet table plus internal routes. These devices, however, may be perfectly reasonable and useful when used for BGP routing of some smaller part of the network, such as a confederation-AS representing one of several smaller enterprises that are linked, by a BGP backbone of backbones, or a small enterprise that announces routes to an ISP but only accepts a default route and perhaps a small number of aggregated routes.

A BGP router used only for a network with a single point of entry to the Internet may have a much smaller routing table size (and hence RAM and CPU requirement) than a multihomed network. Even simple multihoming can have modest routing table size. See RFC 4098 for vendor-independent performance parameters for single BGP router convergence in the control plane. The actual amount of memory required in a BGP router depends on the amount of BGP information exchanged with other BGP speakers, and the way in which the particular router stores BGP information. The router may have to keep more than one copy of a route, so it can manage different policies for route advertising and acceptance to a specific neighboring AS. The term view is often used for these different policy relationships on a running router.

If one router implementation takes more memory per route than another implementation, this may be a legitimate design choice, trading processing speed against memory. A full BGP table as of November 2011 is in excess of 370,000 prefixes. Large ISPs may add another 50% for internal and customer routes. Again depending on implementation, separate tables may be kept for each view of a different peer AS.

Free and open source implementations

- Bird Internet routing daemon, a GPL routing package for Unix-like systems.
- GNU Zebra, a GPL routing suite supporting BGP4.
- OpenBGPD, a BSD licensed implementation by the OpenBSD team.
- Quagga, a fork of GNU Zebra for Unix-like systems.
- XORP, the eXtensible Open Router Platform, a BSD licensed suite of routing protocols.
- VNE [16], a C# software library implementing BGP

Simulators

- BGPviz [17], a Flash application that presents a graphical visualization of BGP routes and updates for any real AS on the Internet
- SSFnet [18], SSFnet network simulator includes a BGP implementation developed by BJ Premore
- C-BGP [19], a BGP simulator able to perform large-scale simulation trying to model the ASes of the Internet or modelling ASes as large as Tier-1.[20]
- BGP++ [21], a patch integrating GNU Zebra software on ns-2 and GTNetS network simulators
- ns-BGP [22], a BGP extension for ns-2 simulator based on the SSFnet implementation
- NetViews [23], a Java application that monitors and visualizes BGP activity in real time.

Test equipment

Systems for testing BGP conformance, load or stress performance come from vendors such as:

- Ixia
- Spirent Communications
- Agilent Technologies

References

[1] Capabilities Advertisement with BGP-4 (http://www.ietf.org/rfc/rfc2842.txt), RFC 2842, R. Chandra & J. Scudder,May 2000

[2] Multiprotocol Extensions for BGP-4 (http://www.ietf.org/rfc/rfc2858.txt), RFC 2858, T. Bates *et al.*,June 2000

[3] BGP/MPLS VPNs. (http://www.ietf.org/rfc/rfc2547.txt), RFC 2547, E. Rosen and Y. Rekhter,April 2004

[4] IANA registry for BGP Extended Communities Types (http://www.iana.org/assignments/bgp-extended-communities), IANA,2008

[5] IETF drafts on BGP signalled QoS (http://www.bgp-qos.org/forum/viewforum.php?f=6), Thomas Knoll,2008

[6] BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP) (http://www.ietf.org/rfc/rfc4456.txt), RFC 4456, T. Bates *et al.*, April 2006

[7] Autonomous System Confederations for BGP (http://www.ietf.org/rfc/rfc5065.txt), RFC 5065, P. Traina *et al.*, February 2001

[8] Border Gateway Protocol (BGP) Persistent Route Oscillation Condition (http://www.ietf.org/rfc/rfc3345.txt), RFC 3345, D. McPherson *et al.*, August 2002

[9] Terminology for Benchmarking BGP Device Convergence in the Control Plane (http://www.ietf.org/rfc/rfc4098.txt), RFC 4098, H. Berkowitz *et al.*, June 2005

[10] http://www.ietf.org/rfc/rfc2439.txt

[11] Route Flap Damping Exacerbates Internet Routing Convergence (http://www.sigcomm.org/sigcomm2002/papers/routedampening.pdf)

[12] Zhang, Beichuan; Pei Dan, Daniel Massey, Lixia Zhang (June 2005). "Timer Interaction in Route Flap Damping" (http://www.cs.arizona.edu/~bzhang/paper/05-icdcs-dtimer.pdf). *IEEE 25th International Conference on Distributed Computing Systems*. . Retrieved 2006-09-26. "We show that the current damping design leads to the intended behavior only under persistent route flapping. When the number of flaps is small, the global routing dynamics deviates significantly from the expected behavior with a longer convergence delay."

[13] http://www.ripe.net/docs/routeflap-damping.html

[14] http://tools.ietf.org/internet-drafts/draft-li-bgp-stability-01.txt

[15] BGP Routing Table Analysis Reports (http://bgp.potaroo.net)

[16] http://wiki.ucis.nl/VNE

[17] http://www.ris.ripe.net/bgpviz/

[18] http://www.ssfnet.org/homePage.html

[19] http://cbgp.info.ucl.ac.be

[20] Quoitin, Bruno; Steve Uhlig (November 2005). "Modeling the Routing of an Autonomous System with C-BGP" (http://inl.info.ucl.ac.be/publications/modeling-routing-autonomous-system-c-). *IEEE Network Magazine* **19** (6). .
[21] http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/
[22] http://www.ensc.sfu.ca/~ljilja/cnl/projects/BGP/
[23] http://netlab.cs.memphis.edu/projects_netviews.html

Further reading

- Chapter "Border Gateway Protocol (BGP)" (http://docwiki.cisco.com/wiki/Border_Gateway_Protocol) in the Cisco "IOS Technology Handbook"

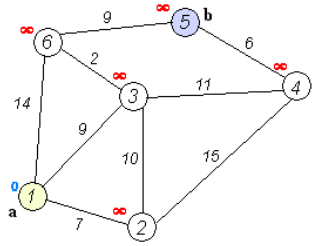
External links

- Cyclops (http://cyclops.cs.ucla.edu/) A BGP network audit tool (prefix hijack, route leakage) by UCLA
- Codenomicon Defensics for BGP (http://www.codenomicon.com/products/bgp.shtml), a test automation framework for BGP Fuzzing and robustness testing
- LinkRank (http://linkrank.cs.ucla.edu/) A tool for BGP routing visualization by University of California, Los Angeles
- A look at fuzzing BGP for security testing (http://www.breakingpointsystems.com/community/blog/protocolfuzzingBGP)
- BGP Routing Resources (http://www.bgp4.as/) (includes a dedicated section on BGP & ISP Core Security (http://www.bgp4.as/security))
- BGP table statistics (http://bgp.potaroo.net/)
- ASNumber Firefox Extension (http://www.asnumber.networx.ch/) showing the AS number and additional information of the website currently open
- RIPE Routing Information Service (http://www.ripe.net/ris/index.html) collecting over 550 IPv4 and IPv6 BGP feeds at 14 sites around the world
- RIS Looking Glass (http://www.ris.ripe.net/cgi-bin/lg/index.cgi) into the Default Free Routing zone of the Internet
- RISwhois (http://www.ripe.net/projects/ris/tools/riswhois.html) providing IPv4/IPv6 Address to BGP AS Origin Mapping
- RIS BGPlay (http://www.ris.ripe.net/bgplay/) BGP routing visualization tool by Università degli Studi Roma Tre
- Linux Magazine: Demystifying BGP (http://www.linux-mag.com/id/1353/) (Good, Detailed BGP explanation; requires registration)
- BGP config generator (http://www.oriontechnologysolutions.com/administration/basic-bgp-config-generator/) Free web-based tool to generate a Cisco/Quagga BGP configuration
- Some important BGP RFCs
  - RFC 4271, A Border Gateway Protocol 4 (BGP-4)
  - RFC 4456, BGP Route Reflection - An Alternative to Full Mesh Internal BGP (IBGP)
  - RFC 4278, Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification
  - RFC 4277, Experience with the BGP-4 Protocol
  - RFC 4276, BGP-4 Implementation Report
  - RFC 4275, BGP-4 MIB Implementation Survey
  - RFC 4274, BGP-4 Protocol Analysis
  - RFC 4273, Definitions of Managed Objects for BGP-4
  - RFC 4272, BGP Security Vulnerabilities Analysis
  - RFC 3392, Capabilities Advertisement with BGP-4

  - RFC 5065, Autonomous System Confederations for BGP
  - RFC 2918, Route Refresh Capability for BGP-4
  - RFC 1772, Application of the Border Gateway Protocol in the Internet Protocol (BGP-4) using SMIv2
  - RFC 4893, BGP Support for Four-octet AS Number Space
  - RFC 2439, BGP Route Flap Damping
  - RFC 4760, Multiprotocol Extensions for BGP-4
- Obsolete RFCs
  - RFC 2796, Obsolete - BGP Route Reflection - An Alternative to Full Mesh IBGP
  - RFC 3065, Obsolete - Autonomous System Confederations for BGP
  - RFC 1965, Obsolete - Autonomous System Confederations for BGP
  - RFC 1771, Obsolete - A Border Gateway Protocol 4 (BGP-4)
  - RFC 1657, Obsolete - Definitions of Managed Objects for the Fourth Version of the Border Gateway
  - RFC 1655, Obsolete - Application of the Border Gateway Protocol in the Internet
  - RFC 1654, Obsolete - A Border Gateway Protocol 4 (BGP-4)
  - RFC 1105, Obsolete - Border Gateway Protocol (BGP)
  - RFC 2858, Obsolete - Multiprotocol Extensions for BGP-4
- BGP Interactions at Router Startup Described as a Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/bgp_startup.pdf) (PDF)
- BGP Protocol Service Level Traffic Variations [[Mu Dynamics (http://www.mudynamics.com/products/protocol.html)]]
- BGP Route Reflection Troubleshooting (http://blog.ipexpert.com/2009/12/21/bgp-route-reflection-â-troubleshooting/)

Dijkstra's algorithm

Dijkstra's algorithm



Dijkstra's algorithm runtime

Class	Search algorithm
Data structure	Graph
Worst case performance	$O(\|E\| + \|V\| \log \|V\|)$

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959,[1] [2] is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path first is widely used in network routing protocols, most notably IS-IS and OSPF (Open Shortest Path First).

Dijkstra's original algorithm does not use a min-priority queue and runs in $O(|V|^2)$. The idea of this algorithm is also given in (Leyzorek et al. 1957). The common implementation based on a min-priority queue implemented by a Fibonacci heap and running in $O(|E| + |V| \log |V|)$ is due to (Fredman & Tarjan 1984). This is asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded nonnegative weights. (For an overview of earlier shortest path algorithms and later improvements and adaptations, see: Single-source shortest-paths algorithms for directed graphs with nonnegative weights.)

Algorithm

Let the node at which we are starting be called the **initial node**. Let the **distance of node Y** be the distance from the **initial node** to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

Illustration of Dijkstra's algorithm search for finding path from a start node to a goal node in a robot motion planning problem. Open nodes represent the "tentative" set. Filled nodes are visited ones, with color representing the distance: the greener, the further. Nodes in all the different directions are explored uniformly, appearing as a more-or-less circular wavefront as Dijkstra's algorithm uses a heuristic identically equal to 0.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the *unvisited set* consisting of all the nodes except the initial node.
3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded distance, then overwrite that distance. Even though a neighbor has been examined, it is not marked as *visited* at this time, and it remains in the *unvisited set*.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again; its distance recorded now is final and minimal.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal), then stop. The algorithm has finished.
6. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step 3.

Description

> ***Note:*** *For ease of understanding, this discussion uses the terms* ***intersection****,* ***road*** *and* ***map*** *— however, formally these terms are* ***vertex****,* ***edge*** *and* ***graph****, respectively.*

Suppose you want to find the shortest path between two intersections on a city map, a starting point and a destination. The order is conceptually simple: to start, mark the distance to every intersection on the map with infinity. This is done not to imply there is an infinite distance, but to note that that intersection has not yet been *visited*; some variants of this method simply leave the intersection unlabeled. Now, at each iteration, select a *current* intersection. For the first iteration the current intersection will be the starting point and the distance to it (the intersection's label) will be zero. For subsequent iterations (after the first) the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find.

From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current intersection, and relabeling the unvisited intersection with this value if it is less than its current value. In effect, the intersection is relabeled if the path to it through the current intersection is shorter than the previously known paths. To facilitate shortest path identification, in pencil, mark the road with an arrow pointing to the relabeled intersection if you label/relabel it, and erase all others pointing to it. After you have updated the distances to each neighboring intersection, mark the current intersection as *visited* and select the unvisited intersection with lowest distance (from the starting point) -- or lowest label—as the current intersection. Nodes marked as visited are labeled with the

shortest path from the starting point to it and will not be revisited or returned to.

Continue this process of updating the neighboring intersections with the shortest distances, then marking the current intersection as visited and moving onto the closest unvisited intersection until you have marked the destination as visited. Once you have marked the destination as visited (as is the case with any visited intersection) you have determined the shortest path to it, from the starting point, and can trace your way back, following the arrows in reverse.

Of note is the fact that this algorithm makes no attempt to direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. In some sense, this algorithm "expands outward" from the starting point, iteratively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path, however it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.

Pseudocode

In the following algorithm, the code `u := vertex in Q with smallest dist[]`, searches for the vertex `u` in the vertex set `Q` that has the least `dist[u]` value. That vertex is removed from the set `Q` and returned to the user. `dist_between(u, v)` calculates the length between the two neighbor-nodes `u` and `v`. The variable `alt` on line 15 is the length of the path from the root node to the neighbor node `v` if it were to go through `u`. If this path is shorter than the current shortest path recorded for `v`, that current path is replaced with this `alt` path. The `previous` array is populated with a pointer to the "next-hop" node on the source graph to get the shortest route to the source.

```
1  function Dijkstra(Graph, source):
2      for each vertex v in Graph:            // Initializations
3          dist[v] := infinity ;              // Unknown distance function from source to v
4          previous[v] := undefined ;         // Previous node in optimal path from source
5      end for ;
6      dist[source] := 0 ;                    // Distance from source to source
7      Q := the set of all nodes in Graph ;   // All nodes in the graph are unoptimized - thus are in Q
8      while Q is not empty:                  // The main loop
9          u := vertex in Q with smallest distance in dist[] ;
10         if dist[u] = infinity:
11             break ;                        // all remaining vertices are inaccessible from source
12         end if ;
13         remove u from Q ;
14         for each neighbor v of u:          // where v has not yet been removed from Q.
15             alt := dist[u] + dist_between(u, v) ;
16             if alt < dist[v]:              // Relax (u,v,a)
17                 dist[v] := alt ;
18                 previous[v] := u ;
19                 decrease-key v in Q;       // Reorder v in the Queue
20             end if ;
21         end for ;
22     end while ;
23     return dist[] ;
24 end Dijkstra.
```

If we are only interested in a shortest path between vertices `source` and `target`, we can terminate the search at line 13 if `u = target`. Now we can read the shortest path from `source` to `target` by iteration:

```
S := empty sequence
u := target
while previous[u] is defined:
    insert u at the beginning of S
    u := previous[u]
end while ;
```

Now sequence `S` is the list of vertices constituting one of the shortest paths from `source` to `target`, or the empty sequence if no path exists.

A more general problem would be to find all the shortest paths between `source` and `target` (there might be several different ones of the same length). Then instead of storing only a single node in each entry of `previous[]` we would store all nodes satisfying the relaxation condition. For example, if both `r` and `source` connect to `target` and both of them lie on different shortest paths through `target` (because the edge cost is the same in both cases), then we would add both `r` and `source` to `previous[target]`. When the algorithm completes, `previous[]` data structure will actually describe a graph that is a subset of the original graph with some edges removed. Its key property will be that if the algorithm was run with some starting node, then every path from that node to any other node in the new graph will be the shortest path between those nodes in the original graph, and all paths of that length from the original graph will be present in the new graph. Then to actually find all these short paths between two given nodes we would use a path finding algorithm on the new graph, such as depth-first search.

Running time

An upper bound of the running time of Dijkstra's algorithm on a graph with edges E and vertices V can be expressed as a function of $|E|$ and $|V|$ using Big-O notation.

For any implementation of vertex set Q the running time is $O(|E| \cdot dk_Q + |V| \cdot em_Q)$, where dk_Q and em_Q are times needed to perform decrease key and extract minimum operations in set Q, respectively.

The simplest implementation of the Dijkstra's algorithm stores vertices of set Q in an ordinary linked list or array, and extract minimum from Q is simply a linear search through all vertices in Q. In this case, the running time is $O(|V|^2 + |E|) = O(|V|^2)$.

For sparse graphs, that is, graphs with far fewer than $O(|V|^2)$ edges, Dijkstra's algorithm can be implemented more efficiently by storing the graph in the form of adjacency lists and using a binary heap, pairing heap, or Fibonacci heap as a priority queue to implement extracting minimum efficiently. With a binary heap, the algorithm requires $O((|E| + |V|) \log |V|)$ time (which is dominated by O(| E | log | V |), assuming the graph is connected). To avoid O(|V|) look-up in decrease-key step on a vanilla binary heap, it is necessary to maintain a supplementary index mapping each vertex to the heap's index (and keep it up to date as priority queue Q changes), making it take only $O(log|V|)$ time instead. The Fibonacci heap improves this to $O(|E| + |V| \log |V|)$.

Note that for Directed acyclic graphs, it is possible to find shortest paths from a given starting vertex in linear time, by processing the vertices in a topological order, and calculating the path length for each vertex to be the minimum length obtained via any of its incoming edges.[3]

Related problems and algorithms

The functionality of Dijkstra's original algorithm can be extended with a variety of modifications. For example, sometimes it is desirable to present solutions which are less than mathematically optimal. To obtain a ranked list of less-than-optimal solutions, the optimal solution is first calculated. A single edge appearing in the optimal solution is removed from the graph, and the optimum solution to this new graph is calculated. Each edge of the original solution is suppressed in turn and a new shortest-path calculated. The secondary solutions are then ranked and presented after the first optimal solution.

Dijkstra's algorithm is usually the working principle behind link-state routing protocols, OSPF and IS-IS being the most common ones.

Unlike Dijkstra's algorithm, the Bellman-Ford algorithm can be used on graphs with negative edge weights, as long as the graph contains no negative cycle reachable from the source vertex *s*. (The presence of such cycles means there is no shortest path, since the total weight becomes lower each time the cycle is traversed.)

The A* algorithm is a generalization of Dijkstra's algorithm that cuts down on the size of the subgraph that must be explored, if additional information is available that provides a lower bound on the "distance" to the target. This approach can be viewed from the perspective of linear programming: there is a natural linear program for computing shortest paths, and solutions to its dual linear program are feasible if and only if they form a consistent heuristic (speaking roughly, since the sign conventions differ from place to place in the literature). This feasible dual / consistent heuristic defines a nonnegative reduced cost and A* is essentially running Dijkstra's algorithm with these reduced costs. If the dual satisfies the weaker condition of admissibility, then A* is instead more akin to the Bellman-Ford algorithm.

The process that underlies Dijkstra's algorithm is similar to the greedy process used in Prim's algorithm. Prim's purpose is to find a minimum spanning tree that connects all nodes in the graph; Dijkstra is concerned with only two nodes. Prim's does not evaluate the total weight of the path from the starting node, only the individual path.

Dynamic programming perspective

From a dynamic programming point of view, Dijkstra's algorithm is a successive approximation scheme that solves the dynamic programming functional equation for the shortest path problem by the **Reaching** method.[4] [5] [6]

In fact, Dijkstra's explanation of the logic behind the algorithm,[7] namely

> **Problem 2.** Find the path of minimum total length between two given nodes P and Q .
>
> We use the fact that, if R is a node on the minimal path from P to Q , knowledge of the latter implies the knowledge of the minimal path from P to R .

is a paraphrasing of Bellman's famous Principle of Optimality in the context of the shortest path problem.

Notes

[1] Dijkstra, Edsger; Thomas J. Misa, Editor (2010-08). "An Interview with Edsger W. Dijkstra". *Communications of the ACM* **53** (8): 41–47. doi:10.1145/1787234.1787249. "What is the shortest way to travel from Rotterdam to Groningen? It is the algorithm for the shortest path which I designed in about 20 minutes. One morning I was shopping with my young fianceé, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path."

[2] Dijkstra 1959

[3] http://www.boost.org/doc/libs/1_44_0/libs/graph/doc/dag_shortest_paths.html

[4] Sniedovich, M. (2006). "Dijkstra's algorithm revisited: the dynamic programming connexion" (http://matwbn.icm.edu.pl/ksiazki/cc/cc35/cc3536.pdf) (PDF). *Journal of Control and Cybernetics* **35** (3): 599–620. . Online version of the paper with interactive computational modules. (http://www.ifors.ms.unimelb.edu.au/tutorial/dijkstra_new/index.html)

[5] Denardo, E.V. (2003). *Dynamic Programming: Models and Applications*. Mineola, NY: Dover Publications. ISBN 978-0486428109.

[6] Sniedovich, M. (2010). *Dynamic Programming: Foundations and Principles*. Francis & Taylor. ISBN 9780824740993.

[7] Dijkstra 1959, p. 270

References

- Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs" (http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf). *Numerische Mathematik* **1**: 269–271. doi:10.1007/BF01386390.
- Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". *Introduction to Algorithms* (Second ed.). MIT Press and McGraw-Hill. pp. 595–601. ISBN 0-262-03293-7.
- Fredman, Michael Lawrence; Tarjan, Robert E. (1984). "Fibonacci heaps and their uses in improved network optimization algorithms" (http://www.computer.org/portal/web/csdl/doi/10.1109/SFCS.1984.715934). *25th Annual Symposium on Foundations of Computer Science* (IEEE): 338–346. doi:10.1109/SFCS.1984.715934.
- Fredman, Michael Lawrence; Tarjan, Robert E. (1987). "Fibonacci heaps and their uses in improved network optimization algorithms" (http://portal.acm.org/citation.cfm?id=28874). *Journal of the Association for Computing Machinery* **34** (3): 596–615. doi:10.1145/28869.28874.
- Zhan, F. Benjamin; Noon, Charles E. (February 1998). "Shortest Path Algorithms: An Evaluation Using Real Road Networks". *Transportation Science* **32** (1): 65–73. doi:10.1287/trsc.32.1.65.
- Leyzorek, M.; Gray, R. S.; Johnson, A. A.; Ladew, W. C.; Meaker, Jr., S. R.; Petry, R. M.; Seitz, R. N. (1957). *Investigation of Model Techniques — First Annual Report — 6 June 1956 — 1 July 1957 — A Study of Model Techniques for Communication Systems*. Cleveland, Ohio: Case Institute of Technology.

External links

- Illustrated explanation of Dijkstra's algorithm (http://www.algolist.com/Dijkstra's_algorithm)
- Oral history interview with Edsger W. Dijkstra (http://www.cbi.umn.edu/oh/display.phtml?id=320), Charles Babbage Institute University of Minnesota, Minneapolis.
- Dijkstra's Algorithm in C# (http://www.codeproject.com/KB/recipes/ShortestPathCalculation.aspx)
- Fast Priority Queue Implementation of Dijkstra's Algorithm in C# (http://www.codeproject.com/KB/recipes/FastHeapDijkstra.aspx)
- Applet by Carla Laffra of Pace University (http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html)
- Animation of Dijkstra's algorithm (http://www.cs.sunysb.edu/~skiena/combinatorica/animations/dijkstra.html)
- Visualization of Dijkstra's Algorithm (http://students.ceid.upatras.gr/~papagel/english/java_docs/minDijk.htm)
- Shortest Path Problem: Dijkstra's Algorithm (http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml)
- Dijkstra's Algorithm Applet (http://www.unf.edu/~wkloster/foundations/DijkstraApplet/DijkstraApplet.htm)
- Open Source Java Graph package with implementation of Dijkstra's Algorithm (http://code.google.com/p/annas/)
- Java Implementation of Dijkstra's Algorithm (http://algowiki.net/wiki/index.php?title=Dijkstra's_algorithm) on AlgoWiki
- QuickGraph, Graph Data Structures and Algorithms for .NET (http://quickgraph.codeplex.com/)
- Implementation in Boost C++ library (http://www.boost.org/doc/libs/1_43_0/libs/graph/doc/dijkstra_shortest_paths.html)
- Implementation in T-SQL (http://hansolav.net/sql/graphs.html)
- A Java library for path finding with Dijkstra's Algorithm and example applet (http://www.stackframe.com/software/PathFinder)
- A MATLAB program for Dijkstra's algorithm (http://www.mathworks.com/matlabcentral/fileexchange/20025-advanced-dijkstras-minimum-path-algorithm)

Distance-vector routing protocol

In computer communication theory relating to packet-switched networks, a **distance-vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman-Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

The term *distance vector* refers to the fact that the protocol manipulates *vectors* (arrays) of distances to other nodes in the network.

Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which or interface to which a packet should be forwarded.
2. Distance from its destination.

Examples of distance-vector routing protocols include RIPv1 and RIPv2 and IGRP. EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

Method

The methods used to calculate the best path for a network are different between different routing protocols but the fundamental features of distance-vector algorithms are the same across all DV based protocols.

Distance Vector means that Routers are advertised as vector of distance and Direction. Direction is simply next hop address and exit interface and Distance means hop count.

Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination.

As the name suggests the DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as ‘routing by rumor’ because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

Limitations

Count-to-infinity problem

The Bellman-Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like as A-B-C-D-E-F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down - B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down - so it tells B that A is only two jumps from C (C to B to A) , which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the "Relax property" of Bellman-Ford).

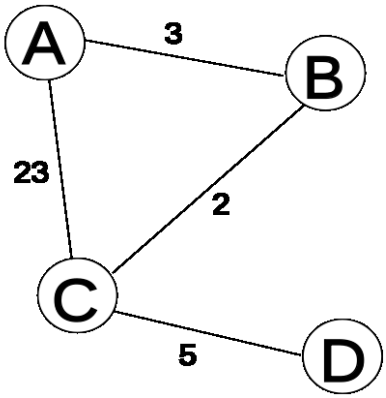
Workarounds and solutions

RIP uses the Split Horizon with Poison Reverse technique to reduce the chance of forming loops and uses a maximum number of hops to counter the 'count-to-infinity' problem. These measures avoid the formation of routing loops in some, but not all, cases. The addition of a *hold time* (refusing route updates for a few minutes after a route retraction) avoids loop formation in virtually all cases, but causes a significant increase in convergence times.

More recently, a number of loop-free distance vector protocols have been developed — notable examples are EIGRP, DSDV and Babel. These avoid loop formation in all cases, but suffer from increased complexity, and their deployment has been slowed down by the success of link-state routing protocols such as OSPF.

Example

In this network we have 4 routers A, B, C, and D:



We shall mark the current time (or iteration) in the algorithm with T, and shall begin (at time 0, or T=0) by creating distance matrices for each router to its immediate neighbors. As we build the routing tables below, the shortest path is highlighted with the color green, a new shortest path is highlighted with the color yellow. Note that all info is along the diagonal.

T=0

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

from D	via A	via B	via C	via D
to A				
to B				
to C			5	
to D				

At this point, all the routers (A,B,C,D) have new "shortest-paths" for their DV (the list of distances that are from them to another router via a neighbor). They each broadcast this new DV to all their neighbors: A to B and C, B to C and A, C to A, B, and D, and D to C. As each of these neighbors receives this information, they now recalculate the shortest path using it.

For example: A receives a DV from C that tells A there is a path via C to D, with a distance (or cost) of 5. Since the current "shortest-path" to C is 23, then A knows it has a path to D that costs 23+5=28. As there are no other shorter paths that A knows about, it puts this as its current estimate for the shortest-path from itself (A) to D, via C.

T=1

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

Again, all the routers have gained in the last iteration (at T=1) new "shortest-paths", so they all broadcast their DVs to their neighbors; This prompts each neighbor to re-calculate their shortest distances again.

For instance: A receives a DV from B that tells A there is a path via B to D, with a distance (or cost) of 7. Since the current "shortest-path" to B is 3, then A knows it has a path to D that costs 7+3=10. This path to D of length 10 (via B) is shorter than the existing "shortest-path" to D of length 28 (via C), so it becomes the new "shortest-path" to D.

T=2

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

This time, only routers A and D have new shortest-paths for their DVs. So they broadcast their new DVs to their neighbors: A broadcasts to B and C, and D broadcasts to C. This causes each of the neighbors receiving the new DVs to re-calculate their shortest paths. However, since the information from the DVs doesn't yield any shorter paths than they already have in their routing tables, then there are no changes to the routing tables.

T=3

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

None of the routers have any new shortest-paths to broadcast. Therefore, none of the routers *receive* any new information that might change their routing tables. So the algorithm comes to a stop.

References

- "RFC1058 - Routing Information Protocol", C. Hedrick, Internet Engineering Task Force, June 1988 [3]
- "RFC1723 - RIP Version 2 Carrying Additional Information", G. Malkin, Internet Engineering Task Force, November, 1994 [1]
- "RFC2453 - RIP Version 2", G. Malkin, Internet Engineering Task Force, November, 1998 [2]
- "A Path-Finding Algorithm for Loop-Free Routing*, J.J. Garcia-Luna-Aceves and S. Murthy, IEEE/ACM Transactions on Networking, February 1997*
- "Detection of Invalid Routing Announcements in the RIP Protocol", D. Pei, D. Massey, and L. Zhang, , IEEE Global Communications Conference (Globecom), December, 2003

Further reading

- Section "Link-State Versus Distance Vector" [3] in the Chapter "Routing Basics" in the Cisco "Internetworking Technology Handbook"

External links

- A Java applet implementing Distance Vector algorithm for pedagogical purposes [4]

References

[1] http://www.ietf.org/rfc/rfc1723.txt
[2] http://www.ietf.org/rfc/rfc2453.txt
[3] http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector
[4] http://www.mathiaz.com/routage/

Enhanced Interior Gateway Routing Protocol

Enhanced Interior Gateway Routing Protocol - (**EIGRP**) is a Cisco proprietary routing protocol loosely based on their original IGRP. EIGRP is an advanced distance-vector routing protocol, with optimizations to minimize both the routing instability incurred after topology changes, as well as the use of bandwidth and processing power in the router. Routers that support EIGRP will automatically redistribute route information to IGRP neighbors by converting the 32 bit EIGRP metric to the 24 bit IGRP metric. Most of the routing optimizations are based on the Diffusing Update Algorithm (DUAL) work from SRI, which guarantees loop-free operation and provides a mechanism for fast convergence.

Basic operation

EIGRP stores data in three tables:

- Neighbor Table: Stores data about the neighboring routers, i.e. those directly accessible through directly connected interfaces.
- Topology Table: Confusingly named, this table does not store an overview of the complete network topology; rather, it effectively contains only the aggregation of the routing tables gathered from all directly connected neighbors. This table contains a list of destination networks in the EIGRP-routed network together with their respective metrics. Also for every destination, a successor and a feasible successor are identified and stored in the table if they exist. Every destination in the topology table can be marked either as "Passive", which is the state when the routing has stabilized and the router knows the route to the destination, or "Active" when the topology has changed and the router is in the process of (actively) updating its route to that destination.
- Routing table: Stores the actual routes to all destinations; the routing table is populated from the topology table with every destination network that has its successor and optionally feasible successor identified (if unequal-cost load-balancing is enabled using the variance command). The successors and feasible successors serve as the next hop routers for these destinations.

Unlike most other distance vector protocols, EIGRP does not rely on periodic route dumps in order to maintain its topology table. Routing information is exchanged only upon the establishment of new neighbor adjacencies, after which only changes are sent. Also, it uses route tagging.

EIGRP Composite and Vector metrics

EIGRP associates six (6) different vector metrics with each route and considers only four (4) of the vector metrics in computing the Composite metric:

```
Router>show ip eigrp topology 10.0.0.1 255.255.255.255
IP-EIGRP topology entry for 10.0.0.1/32
  State is Passive, Query origin flag is 1, 1 Successor(s) , FD is 40640000
  Routing Descriptor Blocks:
  10.0.0.1 (Serial0/0/0) , from 10.0.0.1, Send flag is 0x0
      Composite metric is (40640000/128256) , Route is Internal
      Vector metric:
        Minimum bandwidth is 64 Kbit
        Total delay is 25000 microseconds
        Reliability is 255/255
        Load is 197/255
        Minimum MTU is 576
```

```
        Hop count is 1
```

Bandwidth

- Minimum Bandwidth (in kilobits per second) along the path from router to destination network

Load

- Load (number in range 1 to 255; 255 being saturated)

Delay

- Total Delay (in 10s of microseconds) along the path from router to destination network

Reliability

- Reliability (number in range 1 to 255; 255 being the most reliable)

MTU

- Minimum path Maximum Transmission Unit (MTU) (never used in the metric calculation)

Hop Count

- Number of routers a packet passes through when routing to a remote network, used to limit the EIGRP AS.

The K Values There are five (5) K values used in the Composite metric calculation - K1 through K5. The K values only act as multipliers or modifiers in the composite metric calculation. K1 is not equal to Bandwidth, etc.

By default, only total delay and minimum bandwidth are considered when EIGRP is started on a router, but an administrator can enable or disable all the K values as needed to consider the other Vector metrics.

For the purposes of comparing routes, these are combined together in a weighted formula to produce a single overall metric:

$$\left[\left(K_1 \cdot \text{Bandwidth} + \frac{K_2 \cdot \text{Bandwidth}}{256 - \text{Load}} + K_3 \cdot \text{Delay}\right) \cdot \frac{K_5}{K_4 + \text{Reliability}}\right] \cdot 256$$

where the various constants (K_1 through K_5) can be set by the user to produce varying behaviors. An important and totally non-obvious fact is that if K_5 is set to zero, the term $\frac{K_5}{K_4+\text{Reliability}}$ **is not used (i.e. taken as 1)**.

The default is for K_1 and K_3 to be set to 1, and the rest to zero, effectively reducing the above formula to (Bandwidth + Delay) * 256.

Obviously, these constants must be set to the same value on all routers in an EIGRP system, or permanent routing loops will probably result. Cisco routers running EIGRP will not form an EIGRP adjacency and will complain about K-values mismatch until these values are identical on these routers.

EIGRP scales Bandwidth and Delay metrics with following calculations:

Bandwidth for EIGRP = 10^7 / Interface Bandwidth

Delay for EIGRP = Interface Delay / 10

On Cisco routers, the interface bandwidth is a configurable static parameter expressed in kilobits per second (setting this only affects metric calculation and not actual line bandwidth). Dividing a value of 10^7 kbit/s (i.e. 10 Gbit/s) by the interface bandwidth statement yields a value that is used in the weighted formula. Analogously, the interface delay is a configurable static parameter expressed in microseconds. Dividing this interface delay value by 10 yields a delay in units of tens of microseconds that is used in the weighted formula.

IGRP uses the same basic formula for computing the overall metric, the only difference is that in IGRP, the formula does not contain the scaling factor of 256. In fact, this scaling factor was introduced as a simple means to facilitate backward compatility between EIGRP and IGRP: In IGRP, the overall metric is a 24-bit value while EIGRP uses a 32-bit value to express this metric. By multiplying a 24-bit value with the factor of 256 (effectively bit-shifting it 8 bits to the left), the value is extended into 32 bits, and vice versa. This way, redistributing information between EIGRP and IGRP involves simply dividing or multiplying the metric value by a factor of 256, which is done

automatically.

EIGRP also maintains a hop count for every route, however, the hop count is not used in metric calculation. It is only verified against a predefined maximum on an EIGRP router (by default it is set to 100 and can be changed to any value between 1 and 255). Routes having a hop count higher than the maximum will be advertised as unreachable by an EIGRP router.

Successor

A successor for a particular destination is a next hop router that satisfies these two conditions:

- it provides the least distance to that destination
- it is guaranteed not to be a part of some routing loop

The first condition can be satisfied by comparing metrics from all neighboring routers that advertise that particular destination, increasing the metrics by the cost of the link to that respective neighbor, and selecting the neighbor that yields the least total distance. The second condition can be satisfied by testing a so-called Feasibility Condition for every neighbor advertising that destination. There can be multiple successors for a destination, depending on the actual topology.

The successors for a destination are recorded in the topology table and afterwards they are used to populate the routing table as next-hops for that destination.

Feasible Successor

A feasible successor for a particular destination is a next hop router that satisfies this condition:

- it is guaranteed not to be a part of some routing loop

This condition is also verified by testing the Feasibility Condition.

Thus, every successor is also a feasible successor. However, in most references about EIGRP the term "feasible successor" is used to denote only those routers which provide a loop-free path but which are not successors (i.e. they do not provide the least distance). From this point of view, for a reachable destination there is always at least one successor, however, there might not be any feasible successors.

A feasible successor provides a working route to the same destination, although with a higher distance. At any time, a router can send a packet to a destination marked "Passive" through any of its successors or feasible successors without alerting them in the first place, and this packet will be delivered properly. Feasible successors are also recorded in the topology table.

The feasible successor effectively provides a backup route in the case that existing successors die. Also, when performing unequal-cost load-balancing (balancing the network traffic in inverse proportion to the cost of the routes), the feasible successors are used as next hops in the routing table for the load-balanced destination.

By default, the total count of successors and feasible successors for a destination stored in the routing table is limited to four. This limit can be changed in the range from 1 to 6. In more recent versions of Cisco IOS (e.g. 12.4), this range is between 1 and 16.

Active and Passive State

A destination in the topology table can be marked either as Passive or Active. A Passive state is a state when the router has identified the successor(s) for the destination. The destination changes to Active state when current successor no longer satisfies the Feasibility Condition and there are no feasible successors identified for that destination (i.e. no backup routes are available). The destination changes back from Active to Passive when the router received replies to all queries it has sent to its neighbors. Notice that if a successor stops satisfying the Feasibility Condition but there is at least one feasible successor available, the router will promote a feasible

successor with the lowest total distance (the distance as reported by the feasible successor plus the cost of the link to this neighbor) to a new successor and the destination remains in the Passive state.

Reported Distance and Feasible Distance

Reported Distance (RD) is the total metric along a path to a destination network as advertised by an upstream neighbor.[1] This distance is sometimes also called a Advertised Distance (AD) and is equal to the current lowest total distance through a successor for a neighboring router.

A Feasible Distance (FD) is the lowest known distance from a router to a particular destination. This is the Reported Distance (RD) + the cost to reach the neighboring router from which the RD was sent.[1] It is important to note that this metric represents **the last time the route went from Active to Passive state**. It can be expressed in other words as a historically lowest known distance to a particular destination. While a route remains in Passive state, the FD is updated only if the actual distance to the destination decreases, otherwise it stays at its present value. On the other hand, if a router needs to enter Active state for that destination, the FD will be updated with a new value after the router transitions back from Active to Passive state. This is the only case when the FD can be increased. The transition from Active to Passive state in effect marks the start of a new history for that route.

For example, if the route to a newly discovered destination X went from Active to Passive state with a total distance of 10, the router sets the RD and FD to 10. Later this distance decreases from 10 to 8. The distance remains in the Passive state (because distance decrease never violates the Feasibility Condition) and the router updates the RD and FD to 8. Even later, the distance increases to 12 but in such a way that there is still a valid successor or feasible successor available. In this case, the RD gets updated to 12, however, the FD will remain at the value of 8. Therefore, the values of RD and FD can be different. Finally, the actual successor fails and no other feasible successor is currently identified. Therefore, the router has to transition to Active state and ask its neighbors for a new route to the destination X. Assuming that the newly found path to that destination has a total distance of 10, the router will transition back to Passive state and update both its RD and FD to the new shortest path length, in this case, 10.

Feasibility Condition

The feasibility condition is a sufficient condition for loop freedom in EIGRP-routed network. It is used to select the successors and feasible successors that are guaranteed to be on a loop-free route to a destination. Its simplified formulation is strikingly simple:

If, for a destination, a neighbor router advertises a distance that is strictly lower than our feasible distance, then this neighbor lies on a loop-free route to this destination.

or in other words,

If, for a destination, a neighbor router tells us that it is closer to the destination than we have ever been, then this neighbor lies on a loop-free route to this destination.

In exact terms, every neighbor that satisfies the relation RD < FD for a particular destination is on a loop-free route to that destination.

This condition is also called the Source Node Condition and is one of more equivalent conditions that were proposed and proven by Dr. J. J. Garcia-Luna-Aceves at SRI. The paper proposing the Source Node Condition and the Diffusing Update Algorithm algorithm itself can be found here [2].

It is important to realize that this condition is a sufficient, not a necessary condition. That means that neighbors which satisfy this condition are guaranteed to be on a loop-free path to some destination, however, there may be also other neighbors on a loop-free path which do not satisfy this condition. However, such neighbors do not provide the shortest path to a destination, therefore, not using them does not present any significant impairment of the network functionality. These neighbors will be re-evaluated for possible usage if the router transitions to Active state for that

destination.

EIGRP classification as a distance-vector

In the past, EIGRP was described in various Cisco marketing materials as a balanced hybrid routing protocol, allegedly combining the best features from link-state and distance-vector protocols. This description is not correct from a principal point of view. By definition:

- Distance-vector routing protocols are based on a distributed form of Bellman-Ford algorithm to find shortest paths. They work by exchanging a vector of distances to all destinations known to each node. No further topological information is ever exchanged. Thus, each node knows about all destinations present in the network and it knows the resulting distance to each destination via every of the node's neighbors. However, the node does not have any idea of the actual network topology, nor does the node need it.
- Link-state routing protocols are based on algorithms to find shortest paths in a graph (the most often used algorithm is Dijkstra's algorithm). They work by exchanging a description of each node and its exact connections to its neighbors (in essence, each node describes its adjacencies to neighboring nodes and this information is flooded throughout the network). Therefore, each node knows the exact network topology, i.e. it has a graph representation of the network. Using this graph, each node computes the shortest paths from itself to each available destination.

The EIGRP routers exchange messages that contain information about bandwidth, delay, load, reliability and MTU of the path to each destination as known by the advertising router. Each router uses these parameters to compute the resulting distance to a destination. No further topological information is present in the messages. This principle fully corresponds to the operation of distance-vector protocols. Therefore, EIGRP is in essence a distance-vector protocol.

It is true that EIGRP uses a number of techniques not present in native distance-vector protocols, notably

- the use of explicit hello packets to discover and maintain adjacencies between routers;
- the use of a reliable protocol to transport routing updates;
- the use of a feasibility condition to select a loop-free path;
- the use of diffusing computations to involve the affected part of network into computing a new shortest path.

None of these techniques, however, makes any difference to the basic principles of EIGRP, which exchanges a vector of distances to each known destination network without full knowledge of the network topology, and, as a matter of fact, similar techniques have been used in other distance-vector protocols (notably DSDV, AODV and Babel). While EIGRP is indeed an advanced distance-vector routing protocol, it is not a hybrid protocol.

Other details

EIGRP supports Classless Inter-Domain Routing (CIDR), allowing the use of variable-length subnet masks—one of the protocol's improvements over its predecessor.

EIGRP is not usable in applications where routers need to know the exact network topology (for example, traffic engineering in MPLS).

EIGRP can run separate routing processes for Internet Protocol (IP), IPv6, IPX and AppleTalk through the use of protocol-dependent modules (PDMs). However, this does not facilitate translation between protocols.

Example of setting up EIGRP on a Cisco IOS router for a private network. The 0.0.15.255 wildcard in this example indicates a subnetwork with a maximum of 4094 hosts—it is the bitwise complement of the subnet mask 255.255.240.0. The **no auto-summary** command prevents automatic route summarization on classful boundaries, which would otherwise result in routing loops in discontiguous networks.

```
Router> enable
Router# config terminal
```

```
Router(config)# router eigrp 1
Router(config-router)# network 10.201.96.0 ?
  A.B.C.D  EIGRP wild card bits
  <cr>
Router(config-router)# network 10.201.96.0 0.0.15.255
Router(config-router)# no auto-summary
```

```
Router(config-router)# end
```

References

[1] http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml#feasibleandreported
[2] http://www.soe.ucsc.edu/research/ccrg/publications/jj.dual.ton93.pdf

- Cisco Systems (2005-09-09), *Enhanced Interior Gateway Routing Protocol* (http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml), Document ID 16406, retrieved 2008-04-27.
- Cisco Systems (n.d.), *Internetworking Technology Handbook: Enhanced Interior Gateway Routing Protocol (EIGRP)* (http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Enhanced_IGRP.html), retrieved 2008-04-27.
- Cisco Systems (2005-08-10), *Introduction to EIGRP* (http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080093f07.shtml), Document ID 13669, retrieved 2008-04-27.
- Lammle, Todd (2007), *CCNA Cisco Certified Network Associate Study Guide* (Sixth ed.), Indianapolis, Indiana: Wiley Publishing, ISBN 978-0-470-11008-9.

External links

- Cisco IOS IPv6 Configuration Guide, Release 12.4: Implementing EIGRP for IPv6 (http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-eigrp.html)
- IGRP Metric (http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a008009405c.shtml)
- Loop-free Routing Using Diffusing Computations (http://www.soe.ucsc.edu/research/ccrg/publications/jj.dual.ton93.pdf)
- Termination Detection for Diffusing Computations (http://www.cs.utexas.edu/users/EWD/ewd06xx/EWD687a.PDF)

Exterior Gateway Protocol

The **Exterior Gateway Protocol** (**EGP**) is a now obsolete routing protocol for the Internet originally specified in 1982 by Eric C. Rosen of Bolt, Beranek and Newman, and David L. Mills. It was first described in RFC 827 and formally specified in RFC 904 (1984). Not to be confused with EGPs in general (of which EGP and Border Gateway Protocol (BGP) are examples), EGP is a simple reachability protocol, and, unlike modern distance-vector and path-vector protocols, it is limited to tree-like topologies.

During the early days of the Internet, EGP version 3 (EGP3) was used to interconnect autonomous systems. Currently, BGP version 4 is the accepted standard for Internet routing and has essentially replaced the more limited EGP3.

External links

- TCP/IP Exterior Gateway Protocol (EGP) [1] EGP Overview by The TCP/IP Guide

References

[1] http://www.tcpipguide.com/free/t_TCPIPExteriorGatewayProtocolEGP.htm

Flooding (computer networking)

Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link. See flooding algorithm for details.

Advantages

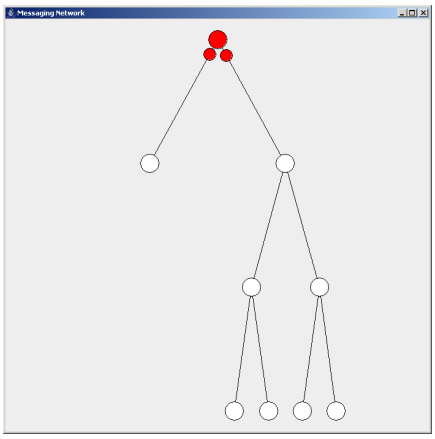
- If a packet can be delivered, it will (probably multiple times).
- Since flooding naturally utilizes every path through the network, it will also use the shortest path.
- This algorithm is very simple to implement.

Problems

- Flooding can be costly in terms of wasted bandwidth and, as in the case of a Ping flood or a Denial of service attack, it can be harmful to the reliability of a computer network.
- Duplicate packets may circulate forever, unless certain precautions are taken:
  - Use a hop count or a time to live count and include it with each packet. This value should take into account the number of nodes that a packet may have to pass through on the way to its destination.
  - Have each node keep track of every packet seen and only forward each packet once.

Flooding algorithm

A **flooding algorithm** is an algorithm for distributing material to every part of a connected network. The name derives from the concept of inundation by a flood.



flooding algorithm

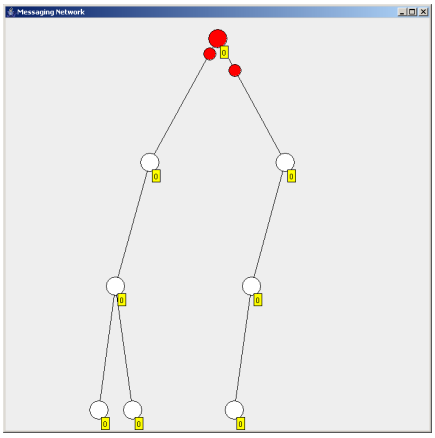
Flooding algorithms are used in systems such as Usenet and peer-to-peer file sharing systems and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks.

There are several variants of flooding algorithm: most work roughly as follows.

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbors except the source node.

This results in every message eventually being delivered to all reachable parts of the network.

Real-world flooding algorithms have to be more complex than this, since precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system.



flooding algorithm with ack messages

Flooding algorithms are also useful for solving many mathematical problems, including maze problems and many problems in graph theory.

Disadvantages of Flooding

There are several disadvantages with this approach to routing. It is very wasteful in terms of the networks total bandwidth. While a message may only have one destination it has to be sent to every host. This increases the maximum load placed upon the network.

Messages can also become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages.

A variant of flooding called *selective flooding* partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don't send every incoming packet on every line but only on those lines which are going approximately in the right direction.

Advantages of Flooding

The main advantage of flooding is the increased reliability provided by this routing method. Since the message will be sent at least once to every host it is almost guaranteed to reach its destination. In addition, the message will reach the host through the shortest possible path.

Examples

- Ricochet [1] A flooding networking app written in Java

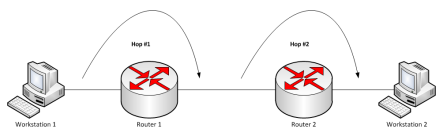
References

[1] http://ricochet.wiki.sourceforge.net/

Hop count

In computer networking, **hop count** refers to the number of routers through which data must pass between source and destination.[1] Each router along the data path constitutes a hop, as the data is moved from one Layer 3 network to another. Hop count is therefore a basic measurement of distance in an network.

By itself, hop count is not a useful metric for determining the optimum network path, as it does not take into consideration the speed, load, reliability, or latency of any particular hop, but merely the total count. Nevertheless, some routing protocols such as RIP use hop count as their sole metric.[2]



An illustration of hops in a network. The hop count between the computers in this case is 2.

Hop counts are often useful to find faults in a network, or to discover if routing is indeed correct. Utilities such as ping and traceroute are used in this capacity—by calculating the hop count to a given destination, as well as enumerating each hop along the way.

References

[1] "Hop Count" (http://www.infocellar.com/networks/ip/hop-count.htm). Infocellar.com. .
[2] RFC 1058, *Routing Information Protocol*, C. Hendrik, The Internet Society (June 1988)

Interior gateway protocol

An **interior gateway protocol** (**IGP**) is a routing protocol that is used to exchange routing information within an autonomous system (AS).

In contrast, an Exterior Gateway Protocol (EGP) is for determining network reachability between autonomous systems and makes use of IGPs to resolve routes within an AS.

The interior gateway protocols can be divided into two categories: 1) Distance-vector routing protocol and 2) Link-state routing protocol.

Types of Interior gateway protocols

Distance-vector routing protocol

Distance-vector routing protocols use the Bellman-Ford algorithm. In these protocols, each router does not possess information about the full network topology. It advertises its distance value (DV) calculated to other routers and receives similar advertisements from other routers unless changes are done in local network or by neighbours (Routers). Using these routing advertisements each router populates its routing table. In the next advertisement cycle, a router advertises updated information from its routing table. This process continues until the routing tables of each router converge to stable values.

Some of these protocols have the disadvantage of slow convergence.

Some examples of Distance Vector routing protocol are:

1. Routing Information Protocol (RIP)
2. Interior Gateway Routing Protocol (IGRP)
3. Enhanced Interior Gateway Routing Protocol (EIGRP)

Link-state routing protocol

In the case of Link-state routing protocols, each router possesses information about the complete network topology. Each router then independently calculates the best next hop from it for every possible destination in the network using local information of the topology. The collection of best next hops forms the routing table.

This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbors. In a link-state protocol, the only information passed between the nodes is information used to construct the connectivity maps.

Some examples of Link-State routing protocol are:

1. Open Shortest Path First (OSPF)
2. Intermediate system to intermediate system (IS-IS)

Hybrid routing protocol

Hybrid routing protocols have both the features of distance vector routing protocols & linked state routing protocols. An example of this protocol is EIGRP.

Interior Gateway Routing Protocol

Interior Gateway Routing Protocol (**IGRP**) is a distance vector interior routing protocol (IGP) invented by Cisco. It is used by routers to exchange routing data within an autonomous system.

IGRP is a proprietary protocol. IGRP was created in part to overcome the limitations of RIP (maximum hop count of only 15, and a single routing metric) when used within large networks. IGRP supports multiple metrics for each route, including bandwidth, delay, load, MTU, and reliability; to compare two routes these metrics are combined together into a single metric, using a formula which can be adjusted through the use of pre-set constants. The maximum hop count of IGRP-routed packets is 255 (default 100), and routing updates are broadcast every 90 seconds (by default).[1]

IGRP is considered a classful routing protocol. Because the protocol has no field for a subnet mask, the router assumes that all subnetwork addresses within the same Class A, Class B, or Class C network have the same subnet mask as the subnet mask configured for the interfaces in question. This contrasts with classless routing protocols that can use variable length subnet masks. Classful protocols have become less popular as they are wasteful of IP address space.

Advancement

In order to address the issues of address space and other factors, Cisco created EIGRP (Enhanced Interior Gateway Routing Protocol). EIGRP adds support for VLSM (variable length subnet mask) and adds the Diffusing Update Algorithm (DUAL) in order to improve routing and provide a loopless environment. EIGRP has completely replaced IGRP, making IGRP an obsolete routing protocol. In Cisco IOS versions 12.3 and greater, IGRP is completely unsupported. In the new Cisco CCNA curriculum (version 4), IGRP is mentioned only briefly, as an "obsolete protocol".

References

[1] Cisco Systems (http://www.cisco.com/en/US/docs/ios/12_2/ip/configuration/guide/1cfigrp.html) Configuring IGRP

External links

- Cisco - An Introduction to IGRP (http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a00800c8ae1.shtml)

IS-IS

Intermediate System To Intermediate System (**IS-IS**), is a routing protocol designed to move information efficiently within a computer network, a group of physically connected computers or similar devices. It accomplishes this by determining the best route for datagrams through a packet-switched network. The protocol was defined in ISO/IEC 10589:2002 as an international standard within the Open Systems Interconnection (OSI) reference design. Though originally an ISO standard, the IETF republished the protocol as an Internet Standard in RFC 1142. IS-IS has been called "the de facto standard for large service provider network backbones."[1]

Description

IS-IS *(pronounced "i-s i-s")* is an interior gateway protocol, designed for use within an administrative domain or network. This is in contrast to Exterior Gateway Protocols, primarily Border Gateway Protocol (BGP), which is used for routing between autonomous systems (RFC 1930).

IS-IS is a link-state routing protocol, operating by reliably flooding link state information throughout a network of routers. Each IS-IS router independently builds a database of the network's topology, aggregating the flooded network information. Like the OSPF protocol, IS-IS uses Dijkstra's algorithm for computing the best path through the network. Packets (datagrams) are then forwarded, based on the computed ideal path, through the network to the destination.

History

The IS-IS protocol was developed by Digital Equipment Corporation as part of DECnet Phase V. It was standardized by the ISO in 1992 as ISO 10589 for communication between network devices which are termed Intermediate Systems (as opposed to end systems or hosts) by the ISO. The purpose of IS-IS was to make possible the routing of datagrams using the ISO-developed OSI protocol stack called CLNS.

IS-IS was developed at roughly the same time that the Internet Engineering Task Force IETF was developing a similar protocol called OSPF. IS-IS was later extended to support routing of *datagrams* in the Internet Protocol (IP), the Network Layer protocol of the global Internet. This version of the IS-IS *routing* protocol was then called *Integrated IS-IS* (RFC 1195 [2]).

Comparison with OSPF

Both IS-IS and OSPF are link state protocols, and both use the same Dijkstra algorithm for computing the best path through the network. As a result, they are conceptually similar. Both support variable length subnet masks, can use multicast to discover neighboring routers using *hello packets*, and can support authentication of routing updates.

While OSPF is natively built to route IP and is itself a Layer 3 protocol that runs on top of IP, IS-IS is natively an OSI network layer protocol (it is at the same layer as CLNS). The widespread adoption of IP worldwide may have contributed to OSPF's popularity. IS-IS does not use IP to carry routing information messages. IS-IS is neutral regarding the type of network addresses for which it can route. OSPF, on the other hand, was designed for IPv4. This allowed IS-IS to be easily used to support IPv6. To operate with IPv6 networks, the OSPF protocol was rewritten in OSPF v3 (as specificed in RFC 2740 [3]).

IS-IS routers build a topological representation of the network. This map indicates the subnets which each IS-IS router can reach, and the lowest-cost (shortest) path to a subnet is used to forward traffic.

IS-IS differs from OSPF in the way that "areas" are defined and routed between. IS-IS routers are designated as being: Level 1 (intra-area); Level 2 (inter area); or Level 1-2 (both). Level 2 routers are inter area routers that can only form relationships with other Level 2 routers. Routing information is exchanged between Level 1 routers and

other Level 1 routers, and Level 2 routers only exchange information with other Level 2 routers. Level 1-2 routers exchange information with both levels and are used to connect the inter area routers with the intra area routers. In OSPF, areas are delineated on the interface such that an area border router (ABR) is actually in two or more areas at once, effectively creating the borders between areas inside the ABR, whereas in IS-IS area borders are in between routers, designated as Level 2 or Level 1-2. The result is that an IS-IS router is only ever a part of a single area. IS-IS also does not require Area 0 (Area Zero) to be the backbone area through which all inter-area traffic must pass. The logical view is that OSPF creates something of a spider web or star topology of many areas all attached directly to Area Zero and IS-IS by contrast creates a logical topology of a backbone of Level 2 routers with branches of Level 1-2 and Level 1 routers forming the individual areas.

IS-IS also differs from OSPF in the methods by which it reliably floods topology and topology change information through the network. However, the basic concepts are similar.

OSPF has a larger set of extensions and optional features. However IS-IS is less "chatty" and can scale to support larger networks. Given the same set of resources, IS-IS can support more routers in an area than OSPF. This has contributed to IS-IS as an ISP-scale protocol.

The TCP/IP implementation, known as "Integrated IS-IS" or "Dual IS-IS", is described in RFC 1195.

Related protocols

- Fabric Shortest Path First (FSPF)
- IEEE 802.1aq - Shortest Path Bridging (SPB)
- Transparent Interconnect of Lots of Links (TRILL)

References

[1] Gredler, Hannes; Goraiski, Walter (2005). *The complete IS-IS routing protocol*. Springer. pp. 1. ISBN 1852338229.
[2] http://www.ietf.org/rfc/rfc1195.txt
[3] http://www.ietf.org/rfc/rfc2740.txt

- RFC 1142 - IS-IS protocol specification (IETF) - Note: this is a copy of DP 10589 (Draft Proposal) and differs in many significant details from the final version of ISO/IEC 10589
- RFC 1195 - Use of OSI IS-IS for Routing in TCP/IP and Dual Environments

External links

- ISO/IEC 10589:2002 Second Edition (http://standards.iso.org/ittf/PubliclyAvailableStandards/c030932_ISO_IEC_10589_2002(E).zip)
- OSPF and IS-IS: A Comparative Anatomy (http://www.nanog.org/meetings/nanog19/presentations/katz.ppt) by Dave Katz, Juniper
- Collection of RFCs pertaining to IS-IS (http://www.networksorcery.com/enp/protocol/is-is.htm)
- Configuring integrated IS-IS on Cisco Routers (http://www.cisco.com/en/US/docs/ios/11_3/np1/configuration/guide/1cisis.html)
- IS-IS and OSPF difference discussion (http://www.nada.kth.se/kurser/kth/2D1490/06/hemuppgifter/bhatia-manral-diff-isis-ospf-01.txt.html) (Vishwas Manral, Manav Bhatia and Yasuhiro Ohara)
- Sample isisd.conf file (http://svn.dd-wrt.com:8000/dd-wrt/browser/src/router/quagga/isisd/isisd.conf.sample?rev=7215): used with Quagga

Link-state advertisement

The **link-state advertisement** (LSA) is a basic communication means of the OSPF routing protocol for the Internet Protocol (IP). It communicates the router's local routing topology to all other local routers in the same OSPF area. OSPF is designed for scalability, so some LSAs are not flooded out on all interfaces, but only on those that belong to the appropriate area. In this way detailed information can be kept localized, while summary information is flooded to the rest of the network. The original IPv4-only OSPFv2 and the newer IPv6-compatible OSPFv3 have broadly similar LSA types.

Types

The LSA types defined in OSPF are as follows:

- *Type 1* - Router LSA - the router announces its presence and lists the links to other routers or networks in the same area, together with the metrics to them. Type 1 LSAs are flooded across their own area only. The link-state ID of the type 1 LSA is the originating router ID.
- *Type 2* - Network LSA - the designated router on a broadcast segment (e.g. Ethernet) lists which routers are joined together by the segment. Type 2 LSAs are flooded across their own area only. The link-state ID of the type 2 LSA is the IP interface address of the DR.
- *Type 3* - Summary LSA - an Area Border Router (ABR) takes information it has learned on one of its attached areas and it can summarize it (but not by default) before sending it out on other areas it is connected to. This summarization helps provide scalability by removing detailed topology information for other areas, because their routing information is summarized into just an address prefix and metric. The summarization process can also be configured to remove a lot of detailed address prefixes and replace them with a single summary prefix, also helping scalability. The link-state ID is the destination network number for type 3 LSAs.
- *Type 4* - ASBR-Summary LSA - this is needed because Type 5 External LSAs are flooded to all areas and the detailed next-hop information may not be available in those other areas. This is solved by an Area Border Router flooding the information for the router (i.e. the Autonomous System Boundary Router) where the type 5 originated. The link-state ID is the router ID of the described ASBR for type 4 LSAs.
- *Type 5* - External LSA - these LSAs contain information imported into OSPF from other routing processes. They are flooded to all areas (except stub areas). For "External Type 1" LSAs routing decisions are made by adding the OSPF metric to get to the ASBR and the external metric from there on, while for "External Type 2" LSAs only the external metric is used. The link-state ID of the type 5 LSA is the external network number.
- *Type 6* - Group Membership LSA - this was defined for Multicast extensions to OSPF (MOSPF)[1] , a multicast OSPF routing protocol which was not in general use. MOSPF has been deprecated since OSPFv3[2] and is not currently used. It may be reassigned in the future.
- *Type 7* - Routers in a Not-so-stubby-area (NSSA) do not receive external LSAs from Area Border Routers, but are allowed to send external routing information for redistribution. They use type 7 LSAs to tell the ABRs about these external routes, which the Area Border Router then translates to type 5 external LSAs and floods as normal to the rest of the OSPF network.
- *Type 8* - A link-local only LSA for OSPFv3. A Type 8 LSA is used to give information about link-local addresses and a list of IPv6 addresses on the link. In OSPFv2, however, the Type 8 was originally intended to be used as a so-called External-Attributes-LSA for transit autonomous systems where OSPFv2 could replace the internal Border Gateway Protocol (iBGP). In these networks, the BGP destinations would be carried in LSA Type 5 while their BGP attributes would be inserted into LSA Type 8. Most OSPFv2 implementations never supported this feature.

- *Type 9* - a link-local "opaque" LSA (defined by RFC2370) in OSPFv2 and the Intra-Area-Prefix LSA in OSPFv3. It is the OSPFv3 LSA that contains prefixes for stub and transit networks in the link-state ID.
- *Type 10* - an area-local "opaque" LSA as defined by RFC2370. Opaque LSAs contain information which should be flooded by other routers even if the router is not able to understand the extended information itself. Typically type 10 LSAs are used for traffic engineering extensions to OSPF, flooding extra information about links beyond just their metric, such as link bandwidth and color.
- *Type 11* - an AS "opaque" LSA defined by RFC 5250, which is flooded everywhere except stub areas. This is the opaque equivalent of the type 5 external LSA.[3]

The opaque LSAs, types 9, 10, and 11, are designated for upgrades to OSPF for application-specific purposes. For example, OSPF-TE has traffic engineering extensions to be used by RSVP-TE in Multiprotocol Label Switching (MPLS). Opaque LSAs are used to flood link color and bandwidth information. Standard LSDB flooding mechanisms are used for distribution of opaque LSAs. Each of the three types has a different flooding scope.

For all types of LSAs, there are 20-byte LSA headers. One of the fields of the LSA header is the link-state ID.

Each router link is defined as one of four types: type 1, 2, 3, or 4. The LSA includes a link ID field that identifies, by the network number and mask, the object that this link connects to.

Depending on the type, the link ID has different meanings.

Link type	Description	Link ID	Link Data
1	point-to-point connection to another router	neighboring router ID	IP address of the originating's interface to the network
2	connection to a transit network	IP address of Designated Router	IP address of the originating's interface to the network
3	connection to a stub network	IP network/subnet number	Subnet mask of the interface
4	virtual link	neighboring router ID	IP address of the originating's interface to the network

---

OSPF v3 LSA Packet Header Formats

As per Appendix A.4.2 [4] of RFC 5340, all LSA packets start with a common LSA "20-byte header" as shown below.

Note: These LSA Packet Headers are all preceded by standard "16-byte" OSPF Headers.

Common LSA 20-byte Packet Header

Bit/ Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	LS Age																LS Type															
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															

160	Depending on the "LS Type" details of the contents will vary. (Details of each LS Type are listed below.)
192	
~	
~	
~	
~	...

As per Appendix A.4 [5] of RFC 5340 (OSPFv3 for IPv6) depending upon the LS Type, there are nine major LSA Packet formats as follows (actually eight as one has been deprecated):

LSA Function Code	LS Type	Description
1	0x2001	Router-LSA
2	0x2002	Network-LSA
3	0x2003	Inter-Area-Prefix-LSA Also referred to as Summary-LSA in OSPFv2.
4	0x2004	Inter-Area-Router-LSA Also referred to as ASBR-Summary-LSA in OSPFv2.
5	0x2005	AS-External-LSA Also referred to as External-LSA or AS-External-LSA in OSPFv2.
~~6~~	~~0x2006~~	~~MOSPF-LSA~~ Deprecated in OSPFv3 (may be reassigned) Was referred to as Multicast-OSPF-LSA in OSPFv2.
7	0x2007	NSSA-LSA Also referred to as NSSA-LSA in OSPFv2.
8	0x2008	Link-LSA
9	0x2009	Intra-Area-Prefix-LSA

The nine different formats for each "Type" of LSA packet are listed below (including the deprecated LSA-6):

Type 1: Router LSA Header

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	0	1	1												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	0			Nt	x	V	E	B	Options																							
192	Type								0								Metric															
224	Interface ID																															

256	Neighbor Interface ID		
288	Neighbor Router ID		
320	...		
~	Type	0	Metric
~	Interface ID		
~	Neighbor Interface ID		
~	Neighbor Router ID		
~	...		

Type 2: Network-LSA Header

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	0	1	2												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	0								Options																							
192	Attached Router																															
~	...																															

Type 3: Inter-Area-Prefix-LSA Header

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	0	1	3												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	0								Metric																							
192	PrefixLength								PrefixOptions								0															

224	Address Prefix
256	
288	...

Type 4: Inter-Area-Router-LSA Header

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	0	1	4												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	0								Options																							
192	0								Metric																							
224	Destination Router ID																															

Type 5: AS-External-LSA Header

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	1	0	5												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160						E	F	T	Metric																							
192	PrefixLength								PrefixOptions								Referenced LS Type															
224	Address Prefix ...																															
256																																

288	Forwarding Address (Optional)
~	
~	
~	
~	External Route Tag (Optional)
~	Referenced Link State ID (Optional)

Type 6: MOSPF LSA Header (*Deprecated*)

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
N/A	This LSA Type is deprecated in OSPF v3 RFC 5340 and thus is not shown here.																															
~																																
~																																

Type 7: NSSA-LSA Header *(Same as Type 5 except for the type number field)*

Bit/ Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	LS Age																0	1	0	7												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160						E	F	T	Metric																							
192	PrefixLength								PrefixOptions								Referenced LS Type															
224	Address Prefix ...																															
256																																
288	Forwarding Address (Optional)																															
~																																
~																																
~																																
~	External Route Tag (Optional)																															
~	Referenced Link State ID (Optional)																															

Type 8: Link-LSA Header

Bit/ Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	LS Age																0	0	0	8												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	Rtr Priority								Options																							
192	Link-local Interface Address																															
224																																
256																																
288																																
320	# prefixes																															
352	PrefixLength								PrefixOptions								0															
384	Address Prefix ...																															
~																																
~	...																															
~	PrefixLength								PrefixOptions								0															
~	Address Prefix ...																															
~																																

Type 9: Intra-Area-Prefix LSA Header

Bit/ Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	LS Age																0	0	0	8												
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	# Prefixes																Referenced LS Type															

192	Referenced Link State ID		
224	Referenced Advertising Router		
256	PrefixLength	PrefixOptions	Metric
288	Address Prefix ...		
320			
352	...		
~	PrefixLength	PrefixOptions	Metric
~	Address Prefix ...		
~			

References

[1] http://tools.ietf.org/html/rfc1584
[2] http://tools.ietf.org/html/rfc5340#appendix-A.4.2.1
[3] http://tools.ietf.org/html/rfc5250
[4] http://tools.ietf.org/html/rfc5340#appendix-A.4.2
[5] http://tools.ietf.org/html/rfc5340#appendix-A.4

External links

- Summary of OSPF LSA Types (http://communitystring.com/2008/07/ospf-lsa-types/)

Link-state routing protocol

A **link-state routing protocol** is one of the two main classes of routing protocols used in packet switching networks for computer communications (the other is the distance-vector routing protocol). Examples of link-state routing protocols include OSPF and IS-IS.

The link-state protocol is performed by every *switching node* in the network (i.e. nodes that are prepared to forward packets; in the Internet, these are called routers). The basic concept of link-state routing is that every node constructs a *map* of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical *path* from it to every possible destination in the network. The collection of best paths will then form the node's routing table.

This contrasts with distance-vector routing protocols, which work by having each node share its **routing table** with its neighbors. In a link-state protocol the only information passed between nodes is **connectivity related**.

Link state algorithms are sometimes characterized informally as each router 'telling the world about its neighbors'.

History

What is believed to be the first adaptive routing network of computers, using link-state routing as its heart, was designed and implemented during 1976-77 by a team from Plessey Radar led by Bernard J Harris; the project was for "Wavell" - a system of computer command and control for the British Army.

The first link-state routing concept was published in 1979 by John M. McQuillan (then at Bolt, Beranek and Newman) as a mechanism that would calculate routes more quickly when network conditions changed, and thus lead to more stable routing.

Later work at BBN Technologies showed how to use the link-state technique in a hierarchical system, i.e. one in which the network was divided into areas, so that each switching node does not need a map of the entire network, only the area(s) in which it is included.

The technique was later adapted for use in the contemporary link-state routing protocols IS-IS and OSPF. Cisco literature refers to EIGRP as a "hybrid" protocol, despite the fact it distributes routing tables instead of topology maps. However, it does synchronize routing tables at start up as OSPF does, and sends specific updates only when topology changes occur.

In 2004 Radia Perlman proposed using link-state routing for Layer 2 frame forwarding with devices called Routing Bridges or RBridges. The Internet Engineering Task Force is standardizing the TRILL protocol to accomplish this.

In the 2007 the IEEE started standardization of the use of IS-IS to control Ethernet forwarding with the IEEE 802.1aq Shortest Path Bridging project based on earlier proprietary work Provider Link-State Bridging PLSB.

More recently, this hierarchical technique was applied to wireless mesh networks using the optimized link state routing protocol. Where a connection can have varying quality, the quality of a connection can be used to select better connections. This is used in some routing protocols that use radio frequency transmission.

Distributing maps

This description covers only the simplest configuration; i.e. one with no areas, so that all nodes do have a map of the entire network. The hierarchical case is somewhat more complex; see the various protocol specifications.

As previously mentioned, the first main stage in the link-state algorithm is to give a map of the network to every node. This is done with several simple subsidiary steps.

Determining the neighbors of each node

First, each node needs to determine what other ports it is connected to, over fully working links; it does this using a simple *reachability protocol* which it runs separately with each of its directly connected neighbors.

Distributing the information for the map

Next, each node periodically and in case of connectivity changes makes up a short message, the link-state advertisement, which:

- Identifies the node which is producing it.
- Identifies all the other nodes (either routers or networks) to which it is directly connected.
- Includes a *sequence number*, which increases every time the source node makes up a new version of the message.

This message is then *flooded* throughout the network. As a necessary precursor, each node in the network remembers, for every other node in the network, the sequence number of the last link-state message which it received from that node. With that in hand, the method used is simple.

Starting with the node which originally produced the message, it sends a copy to all of its neighbors. When a link-state advertisement is received at a node, the node looks up the sequence number it has stored for the source of that link-state message. If this message is newer (i.e. has a higher sequence number), it is saved, and a copy is sent in turn to each of that node's neighbors.

This procedure rapidly gets a copy of the latest version of each node's link-state advertisement to every node in the network.

Networks running link state algorithms can also be segmented into hierarchies which limit the scope of route changes. These features mean that link state algorithms scale better to larger networks.

Creating the map

Finally, with the complete set of link-state advertisements (one from each node in the network) in hand, it is obviously easy to produce the graph for the map of the network.

The algorithm simply iterates over the collection of link-state advertisements; for each one, it makes links on the map of the network, from the node which sent that message, to all the nodes which that message indicates are neighbors of the sending node.

No link is considered to have been correctly reported unless the two ends agree; i.e. if one node reports that it is connected to another, but the other node does not report that it is connected to the first, there is a problem, and the link is not included on the map.

Notes about this stage

The link-state message giving information about the neighbors is recomputed, and then flooded throughout the network, whenever there is a change in the connectivity between the node and its neighbors, e.g. when a link fails. Any such change will be detected by the reachability protocol which each node runs with its neighbors.

Calculating the routing table

As initially mentioned, the second main stage in the link-state algorithm is to produce routing tables, by inspecting the maps. This is again done with several steps.

Calculating the shortest paths

Each node independently runs an algorithm over the map to determine the shortest path from itself to every other node in the network; generally some variant of Dijkstra's algorithm is used. This is based around a link cost across each path which includes available bandwidth among other things.

Basically, a node maintains two data structures: a tree containing nodes which are "done", and a list of *candidates*. The algorithm starts with both structures empty; it then adds to the first one the node itself. The variant of a Greedy Algorithm then repetitively does the following:

- All nodes which are connected to the node just added to the tree (excepting any nodes which are already in either the tree or the candidate list) are added to the second (candidate) list.
- Each node in the candidate list is compared to each of the nodes already in the tree. The candidate node which is closest to any of the nodes already in the tree is itself moved into the tree and attached to the appropriate neighbor node. When a node is moved from the candidate list into the tree, it is removed from the candidate list and is not considered in subsequent iterations of the algorithm.

The above two steps are repeated as long as there aren't any nodes left in the candidate list. (When there are none, all the nodes in the network will have been added to the tree.) This procedure ends with the tree containing all the nodes in the network, with the node on which the algorithm is running as the *root* of the tree. The shortest path from that node to any other node is indicated by the list of nodes one traverses to get from the root of the tree, to the desired node in the tree.

Filling the routing table

With the shortest paths in hand, filling in the routing table is trivial.

For any given destination node, the best path for that destination is the node which is the first step from the root node, down the branch in the shortest-path tree which leads toward the desired destination node.

To create the routing table, it is only necessary to walk the tree, remembering the identity of the node at the head of each branch, and filling in the routing table entry for each node one comes across with that identity.

Optimizations to the algorithm

The algorithm described above was made as simple as possible, to aid in ease of understanding. In practice, there are a number of optimizations which are used.

Most importantly, whenever a change in the connectivity map happens, it is necessary to recompute the shortest-path tree, and then recreate the routing table. Work by BBN Technologies discovered how to recompute only that part of the tree which could have been affected by a given change in the map.

Also, the routing table would normally be filled in as the shortest-path tree is computed, instead of making it a separate operation.

Failure modes

If all the nodes are not working from **exactly** the same map, *routing loops* can form. (These are situations in which, in the simplest form, two neighboring nodes each think the other is the best path to a given destination. Any packet headed to that destination arriving at either node will loop between the two, hence the name. Routing loops involving more than two nodes are also possible.)

The reason is fairly simple: since each node computes its shortest-path tree and its routing table without interacting in any way with any other nodes, then if two nodes start with different maps, it is easy to have scenarios in which routing loops are created.

The Optimized Link State Routing Protocol for Mobile ad-hoc Networks

A link-state routing protocol - optimized for mobile ad-hoc networks, which can also be used on other wireless ad-hoc networks - is the *Optimized Link State Routing Protocol* (OLSR).[1] OLSR is proactive, it uses Hello and Topology Control (TC) messages to discover and disseminate link state information into the mobile ad-hoc network. Using Hello messages each node discovers 2-hop neighbor information and elects a set of *multipoint relays* (MPRs). MPRs makes OLSR unique from other link state routing protocols. Individual nodes use the topology information to compute next hop paths regard to all nodes in the network utilising shortest hop forwarding paths.

References

[1] RFC 3626

- John M. McQuillan, Isaac Richer and Eric C. Rosen, *ARPANet Routing Algorithm Improvements*, BBN Report No. 3803, Cambridge, April 1978
- John M. McQuillan, Isaac Richer and Eric C. Rosen, *The New Routing Algorithm for the ARPANet*, IEEE Trans. on Comm., 28(5), pp. 711–719, 1980
- Josh Seeger and Atul Khanna, *Reducing Routing Overhead in a Growing DDN*, MILCOMM '86, IEEE, 1986
- Radia Perlman “Rbridges: Transparent Routing” (http://www.ieee-infocom.org/2004/Papers/26_1.PDF), Infocom 2004.

Further reading

- Section "Link-State Versus Distance Vector" (http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector) in the Chapter "Routing Basics" in the Cisco "Internetworking Technology Handbook"

Metrics (networking)

For other uses, see Metric (disambiguation).

Metrics is a property of a route in computer networking, consisting of any value used by routing algorithms to determine whether one route should perform better than another. The routing table stores only the best possible routes, while link-state or topological databases may store all other information as well. For example, Routing Information Protocol uses hopcount (number of hops) to determine the best possible route.

A Metric can include:

- measuring link utilisation (using SNMP)
- number of hops (hop count)
- speed of the path
- packet loss (router congestion/conditions)
- latency (delay)
- path reliability
- path bandwidth
- throughput [SNMP - query routers]
- load
- MTU

In EIGRP, metrics is represented by an integer from 0 to 4294967295. In Microsoft Windows XP routing it ranges from 1 to 9999.

A Metric can be considered as [1] :

- additive - the total cost of a path is the sum of the costs of individual links along the path,
- concave - the total cost of a path is the minimum of the costs of individual links along the path,
- multiplicative - the total cost of a path is the product of the costs of individual links along the path.

A survey of routing metrics can be found here [2].

Service Level Metrics

Service Level Metrics are concerned with the end user's experience of using the product.

Availability

The availability of a computer network (or an individual service) may be expressed using the notation hh/d/ww. For a 24 hour service, seven days a week, available all year around, this would be expressed 24/7/52 (where the 52 stands for the number of weeks in a year). Service providers usually express that a service will be available for a *percentage* of this time.

To calculate the availability of a service expressed in this format, you need to do the following calculation:

98% availability on 24/7/52

1. Multiply 24 hours per day by 7 days per week by 52 weeks per year = 8736 hours per year
2. Find 98% of the hours per year = 8736 * 98 / 100 = 8561.28 hours guaranteed

You can then deduce how many full hours/days per year the service can be unavailable before the supplier is in breach of their Service Level Agreement. In this example, 8736 (hours) - 8561 (hours) = 175 hours (or around 7.3 days).

Notes

[1] S. D. Rao, C. S. R. Murthy: "Distributed dynamic QoS-aware routing in WDM optical networks", Computer Networks, Volume 48, Issue 4, 15 July 2005, Pages 585-604 (http://dx.doi.org/10.1016/j.comnet.2004.11.003)

[2] http://rainer.baumann.info/public/tik262.pdf

Multipath routing

Multipath routing is the routing technique of using multiple alternative paths through a network, which can yield a variety of benefits such as fault tolerance, increased bandwidth, or improved security. The multiple paths computed might be overlapped, edge-disjointed or node-disjointed with each other. Extensive research has been done on multipath routing techniques, but multipath routing is not yet widely deployed in practice.

Multipath routing in wireless networks

To improve performance or fault tolerance:

CMR (Concurrent Multipath Routing) is often taken to mean simultaneous management and utilization of multiple available paths for the transmission of streams of data emanating from an application or multiple applications. In this form, each stream is assigned a separate path, uniquely to the extent supported by the number of paths available. If there are more streams than available paths, some streams will share paths. This provides better utilization of available bandwidth by creating multiple active transmission queues. It also provides a measure of fault tolerance in that, should a path fail, only the traffic assigned to that path is affected, the other paths continuing to serve their stream flows; there is also, ideally, an alternative path immediately available upon which to continue or restart the interrupted stream.

This method provides better transmission performance and fault tolerance by providing:

· Simultaneous, parallel transport over multiple carriers.

· Load balancing over available assets.

· Avoidance of path discovery when re-assigning an interrupted stream.

Shortcomings of this method are:

· Some applications may be slower in offering traffic to the transport layer, thus starving paths assigned to them, causing underutilization.

· Moving to the alternative path will incur a potentially disruptive period during which the connection is re-established.

A more powerful form of CMR (true CMR) goes beyond merely presenting paths to applications to which they can bind. True CMR aggregates all available paths into a single, virtual path. All applications offer their packets to this virtual path, which is de-muxed at the Network Layer, the packets then being distributed to the actual paths via some method such as round-robin or weighted fair queuing. Should a link or relay node fail, thus invalidating one or more paths, succeeding packets are not directed to that (those) paths. The stream continues uninterrupted, transparently to the application. This method provides significant performance benefits over the former:

· By continually offering packets to all paths, the paths are more fully utilized.

· No matter how many nodes (and thus paths) fail, so long as at least one path constituting the virtual path is still available, all sessions remain connected. This means that no streams need to be restarted from the beginning and no re-connection penalty is incurred.

It is noted that true CMR can, by its nature, cause Out-Of-Order-Delivery (OOOD) of packets, which is severely debilitating for standard TCP. Standard TCP, however, has been exhaustively proven to be inappropriate for use in challenged wireless environments and must, in any case, be augmented by a facility, such as a TCP gateway, that is

designed to meet the challenge. One such gateway tool is SCPS-TP, which, through its Selective Negative Acknowledgement (SNACK) capability, deals successfully with the OOOD problem.

Another important benefit of true CMR, desperately needed in wireless network communications, is its support for enhanced security. Simply put, for an exchange to be compromised, multiple of the routes it traverses must be compromised. The reader is referred to the references in the “To improve network security” section for discussion on this topic.

References

- S.-J. Lee and M. Gerla, “Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks,” Proc. ICC 2001, vol. 10, pp. 3201–3205, June 2001.
- A. Nasipuri, R. Castaneda, and S. R. Das, “Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks,” Mobile Networks and Applications, vol. 6, no. 4, pp. 339–349, Aug. 2001.
- M. K. Marina and S. R. Das “On-Demand Multi Path Distance Vector Routing in Ad Hoc Networks,” Proc. ICNP 2001, pp. 14–23, Nov. 2001.
- A. Tsirigos and Z. J. Haas, “ Multipath Routing in the Presence of Frequent Topological Changes,” IEEE Communications Magazine, vol. 39, no. 11, pp. 132–138, Nov. 2001.
- H. Lim, K. Xu, and M. Gerla, “TCP Performance over Multipath Routing in Mobile Ad Hoc Networks,” Proc. ICC 2003, vol. 2, pp. 1064–1068, May 2003.
- A. Tsirigos and Z. J. Haas, “Analysis of Multipath Routing—Part I: The Effect on the Packet Delivery Ratio,” IEEE Trans. Wireless Communications, vol. 3, no. 1, pp. 138–146, Jan. 2004.
- S. Card, F. Tims, "Concurrent Multipath Routing & Transport in a Mobile Wireless Gateway,"unclassified paper presented in MILCOM 2004 classified session, available upon request from support at www.critical.com.

To improve network security:

- W. Lou and Y. Fang, ""A Multipath Routing Approach for Secure Data Delivery,"" Proc. MILCOM 2001, vol. 2, pp. 1467–1473, Oct. 2001.
- C. K.-L. Lee, X.-H. Lin, and Y.-K. Kwok, “A Multipath Ad Hoc Routing Approach to Combat Wireless Link Insecurity,” Proc. ICC 2003, vol. 1, pp. 448–452, May 2003.
- S. Bouam and J. Ben-Othman, “Data Security in Ad Hoc Networks Using Multipath Routing,” Proc. PIMRC 2003, vol. 2, pp. 1331–1335, Sept. 2003.
- P. Papadimitratos and Z. J. Haas, “Secure Data Transmission in Mobile Ad Hoc Networks,” Proc. ACM WiSe 2003, pp. 41–50, Sept. 2003.
- Zhi Li and Yu-Kwong Kwok, "A New Multipath Routing Approach to Enhancing TCP Security in Ad Hoc Wireless Networks," Proc. ICPP Workshops, pp. 372-379, June 2005.

External links

- Prof. Dijiang Huang's multipath routing bibliography: [1]

References

[1] http://snac.eas.asu.edu/snac/multipath/multipath.html

Open Shortest Path First

Open Shortest Path First (**OSPF**) is an adaptive routing protocol for Internet Protocol (IP) networks. It uses a link state routing algorithm and falls into the group of interior routing protocols, operating within a single autonomous system (AS). It is defined as OSPF Version 2 in RFC 2328 (1998) for IPv4.[1] The updates for IPv6 are specified as OSPF Version 3 in RFC 5340 (2008).[2]

OSPF is perhaps the most widely-used interior gateway protocol (IGP) in large enterprise networks. IS-IS, another link-state routing protocol, is more common in large service provider networks. The most widely-used exterior gateway protocol is the Border Gateway Protocol (BGP), the principal routing protocol between autonomous systems on the Internet.

Overview

OSPF is an interior gateway protocol that routes Internet Protocol (IP) packets solely within a single routing domain (autonomous system). It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer which makes routing decisions based solely on the destination IP address found in IP packets. OSPF was designed to support variable-length subnet masking (VLSM) or Classless Inter-Domain Routing (CIDR) addressing models.

OSPF detects changes in the topology, such as link failures, very quickly and converges on a new loop-free routing structure within seconds. It computes the shortest path tree for each route using a method based on Dijkstra's algorithm, a shortest path first algorithm.

The **link-state** information is maintained on each router as a link-state database (LSDB) which is a tree-image of the entire network topology. Identical copies of the LSDB are periodically updated through flooding on all OSPF routers.

The OSPF routing policies to construct a route table are governed by link cost factors (*external metrics*) associated with each routing interface. Cost factors may be the distance of a router (round-trip time), network throughput of a link, or link availability and reliability, expressed as simple unitless numbers. This provides a dynamic process of traffic load balancing between routes of equal cost.

An OSPF network may be structured, or subdivided, into routing *areas* to simplify administration and optimize traffic and resource utilization. Areas are identified by 32-bit numbers, expressed either simply in decimal, or often in octet-based dot-decimal notation, familiar from IPv4 address notation.

By convention, area 0 (zero) or `0.0.0.0` represents the core or *backbone* region of an OSPF network. The identifications of other areas may be chosen at will; often, administrators select the IP address of a main router in an area as the area's identification. Each additional area must have a direct or virtual connection to the backbone OSPF area. Such connections are maintained by an interconnecting router, known as *area border router* (ABR). An ABR maintains separate link state databases for each area it serves and maintains summarized routes for all areas in the network.

OSPF does not use a TCP/IP transport protocol (UDP, TCP), but is encapsulated directly in IP datagrams with protocol number 89. This is in contrast to other routing protocols, such as the Routing Information Protocol (RIP), or the Border Gateway Protocol (BGP). OSPF handles its own error detection and correction functions.

OSPF uses multicast addressing for route flooding on a broadcast network link. For non-broadcast networks special provisions for configuration facilitate neighbor discovery.[1] OSPF multicast IP packets never traverse IP routers, they never travel more than one hop. OSPF reserves the multicast addresses `224.0.0.5` for IPv4 or `FF02::5` for IPv6 (all SPF/link state routers, also known as `AllSPFRouters`) and `224.0.0.6` for IPv4 or `FF02::6` for IPv6 (all Designated Routers, `AllDRouters`), as specified in RFC 2328[3] and RFC 5340.[4]

For routing multicast IP traffic, OSPF supports the Multicast Open Shortest Path First protocol (MOSPF) as defined in RFC 1584.[5] Neither Cisco nor Juniper Networks include MOSPF in their OSPF implementations. PIM (Protocol Independent Multicast) in conjunction with OSPF or other IGPs, (Interior Gateway Protocol), is widely deployed.

The OSPF protocol, when running on IPv4, can operate securely between routers, optionally using a variety of authentication methods to allow only trusted routers to participate in routing. OSPFv3, running on IPv6, no longer supports protocol-internal authentication. Instead, it relies on IPv6 protocol security (IPsec).

OSPF version 3 introduces modifications to the IPv4 implementation of the protocol.[2] Except for virtual links, all neighbor exchanges use IPv6 link-local addressing exclusively. The IPv6 protocol runs per link, rather than based on the subnet. All IP prefix information has been removed from the link-state advertisements and from the *Hello* discovery packet making OSPFv3 essentially protocol-independent. Despite the expanded IP addressing to 128-bits in IPv6, area and router identifications are still based on 32-bit values.

Neighbor relationships

Routers in the same broadcast domain or at each end of a point-to-point telecommunications link form *adjacencies* when they have detected each other. This detection occurs when a router identifies itself in a *hello* OSPF protocol packet. This is called a *two-way state* and is the most basic relationship. The routers in an Ethernet or frame relay network select a *designated router* (DR) and a *backup designated router* (BDR) which act as a hub to reduce traffic between routers. OSPF uses both unicast and multicast to send "hello packets" and link state updates.

As a link state routing protocol, OSPF establishes and maintains neighbor relationships in order to exchange routing updates with other routers. The neighbor relationship table is called an adjacency database in OSPF. Provided that OSPF is configured correctly, OSPF forms neighbor relationships only with the routers directly connected to it. In order to form a neighbor relationship between two routers, the interfaces used to form the relationship must be in the same area. An interface can only belong to a single area. (A neighbor state simulation [6] shows how neighbor state changes from Down to Full Adjacency progressively with exchanging Hello, DD, Request, Update, and Ack packets).

Area types

An OSPF domain is divided into *areas* that are labeled with 32-bit area identifiers. The area identifiers are commonly, but not always, written in the dot-decimal notation of an IPv4 address. However, they are *not* IP addresses and may duplicate, without conflict, any IPv4 address. The area identifiers for IPv6 implementations of OSPF (OSPFv3) also use 32-bit identifiers written in the same notation. While most OSPF implementations will right-justify an area number written in a format other than dotted decimal format (e.g., area 1), it is wise to always use dotted-decimal formats. Most implementations expand area 1 to the area identifier 0.0.0.1, but some have been known to expand it as 1.0.0.0.

Areas are logical groupings of hosts and networks, including their routers having interfaces connected to any of the included networks. Each area maintains a separate link state database whose information may be summarized towards the rest of the network by the connecting router. Thus, the topology of an area is unknown outside of the area. This reduces the amount of routing traffic between parts of an autonomous system. (An ABR simulation [7] shows how an ABR lets areas know each others' network addresses by flooding Summary LSA.)

Several special area types are defined.

Backbone area

The backbone area (also known as *area 0* or *area 0.0.0.0*) forms the core of an OSPF network. All other areas are connected to it, and inter-area routing happens via routers connected to the backbone area and to their own associated areas. It is the logical and physical structure for the 'OSPF domain' and is attached to all nonzero areas in the OSPF domain. Note that in OSPF the term Autonomous System Boundary Router (ASBR) is historic, in the sense that many OSPF domains can coexist in the same Internet-visible autonomous system, RFC1996 (ASGuidelines 1996, p. 25).[8]

The backbone area is responsible for distributing routing information between nonbackbone areas. The backbone must be contiguous, but it does not need to be physically contiguous; backbone connectivity can be established and maintained through the configuration of virtual links.

All OSPF areas must connect to the backbone area. This connection, however, can be through a virtual link. For example, assume area 0.0.0.1 has a physical connection to area 0.0.0.0. Further assume that area 0.0.0.2 has no direct connection to the backbone, but this area does have a connection to area 0.0.0.1. Area 0.0.0.2 can use a virtual link through the *transit area* 0.0.0.1 to reach the backbone. To be a transit area, an area has to have the transit attribute, so it cannot be stubby in any way.

Stub area

A stub area is an area which does not receive route advertisements external to the autonomous system (AS) and routing from within the area is based entirely on a default route. A Stub Area simulation [9] shows how an ABR deletes type 4, 5 LSAs from internal routers, sends them a default route of 0.0.0.0 and turns itself into a default gateway. This reduces LSDB and routing table size for internal routers.

Modifications to the basic concept of stub areas exist in the *not-so-stubby* area (NSSA). In addition, several other proprietary variations have been implemented by systems vendors, such as the *totally stubby area* (TSA) and the *NSSA not so stubby area*, both an extension in Cisco Systems routing equipment.

Not-so-stubby area

A *not-so-stubby area* (NSSA) is a type of stub area that can import autonomous system external routes and send them to other areas, but still cannot receive AS-external routes from other areas. NSSA is an extension of the stub area feature that allows the injection of external routes in a limited fashion into the stub area. A case study simulates [10] an NSSA getting around the Stub Area problem of not being able to import external addresses. It visualizes the following activities: the ASBR imports external addresses with a type 7 LSA, the ABR converts a type 7 LSA to type 5 and floods it to other areas, the ABR acts as an "ASBR" for other areas.

Proprietary extensions

Several vendors (Cisco, Juniper, Alcatel-Lucent, Huawei, Quagga), now implement below two extensions to stub and NSSA area and although not covered by RFC they are considered by many to be standard features in OSPF implementations.

Totally stubby area

> A *totally stubby area* is similar to a stub area. However, this area does not allow *summary* routes in addition to not having *external* routes, that is, *inter-area* (IA) routes are not summarized into totally stubby areas. The only way for traffic to get routed outside of the area is a default route which is the only Type-3 LSA advertised into the area. When there is only one route out of the area, fewer routing decisions have to be made by the route processor, which lowers system resource utilization.
>
> Occasionally, it is said that a TSA can have only one ABR.[11]

NSSA totally stubby area

An addition to the standard functionality of an NSSA, the *totally stubby NSSA* is an NSSA that takes on the attributes of a TSA, meaning that type 3 and 4 summary routes are not flooded into this type of area. It is also possible to declare an area both totally stubby and not-so-stubby, which means that the area will receive only the default route from area 0.0.0.0, but can also contain an autonomous system boundary router (ASBR) that accepts external routing information and injects it into the local area, and from the local area into area 0.0.0.0.

Redistribution into an NSSA area creates a special type of LSA known as TYPE 7, which can exist only in an NSSA area. An NSSA ASBR generates this LSA, and an NSSA ABR router translates it into type 5 LSA which gets propagated into the OSPF domain.

A newly acquired subsidiary is one example of where it might be suitable for an area to be simultaneously not-so-stubby and totally stubby if the practical place to put an ASBR is on the edge of a totally stubby area. In such a case, the ASBR does send externals into the totally stubby area, and they are available to OSPF speakers within that area. In Cisco's implementation, the external routes can be summarized before injecting them into the totally stubby area. In general, the ASBR should not advertise default into the TSA-NSSA, although this can work with extremely careful design and operation, for the limited special cases in which such an advertisement makes sense.

By declaring the totally stubby area as NSSA, no external routes from the backbone, except the default route, enter the area being discussed. The externals do reach area 0.0.0.0 via the TSA-NSSA, but no routes other than the default route enter the TSA-NSSA. Routers in the TSA-NSSA send all traffic to the ABR, except to routes advertised by the ASBR.

Transit area

A transit area is an area with two or more OSPF border routers and is used to pass network traffic from one adjacent area to another. The transit area does not originate this traffic and is not the destination of such traffic.

Path preference

OSPF uses *path cost* as its basic routing metric, which was defined by the standard not to equate to any standard value such as speed, so the network designer could pick a metric important to the design. In practice, it is determined by the speed (bandwidth) of the interface addressing the given route, although that tends to need network-specific scaling factors now that links faster than 100 Mbit/s are common. Cisco uses a metric like 10^8/bandwidth (the base value, 10^8 by default, can be adjusted). So, a 100Mbit/s link will have a cost of 1, a 10Mbit/s a cost of 10 and so on. But for links faster than 100Mbit/s, the cost would be <1.

Metrics, however, are only directly comparable when of the same type. Four types of metrics are recognized. An intra-area route is always preferred to an External route regardless of metric. In decreasing preference, these types are:

1. Intra-area
2. Inter-area
3. External Type 1, which includes both the external path cost and the sum of internal path costs to the ASBR that advertises the route,
4. External Type 2, the value of which is solely that of the external path cost

Traffic engineering

OSPF-TE is an extension to OSPF extending the expressivity to allow for traffic engineering and use on non-IP networks (RFC 3630).[12] More information about the topology can be exchanged using opaque LSA carrying type-length-value elements. These extensions allow OSPF-TE to run completely out of band of the data plane network. This means that it can also be used on non-IP networks, such as optical networks.

OSPF-TE is used in GMPLS networks as a means to describe the topology over which GMPLS paths can be established. GMPLS uses its own path setup and forwarding protocols, once it has the full network map.

In the Resource Reservation Protocol (RSVP), OSPF-TE is used for recording and flooding RSVP signaled bandwidth reservations for Label switched paths within the link-state database.

Other extensions

RFC 3717 documents work in optical routing for IP, based on "constraint-based" extensions to OSPF and IS-IS.[13]

OSPF router types

OSPF defines the following router types:

- Area border router (ABR)
- Autonomous system boundary router (ASBR)
- Internal router (IR)
- Backbone router (BR)

The router type is an attribute of an OSPF process. A given physical router may have one or more OSPF processes. For example, a router that is connected to more than one area, and which receives routes from a BGP process connected to another AS, is both an area border router and an autonomous system boundary router.

Each router has an identifier, customarily written in the dotted decimal format (e.g., 1.2.3.4) of an IP address. This identifier must be established in every OSPF instance. If not explicitly configured, the highest logical IP address will be duplicated as the router identifier. However, since the router identifier is not an IP address, it does not have to be a part of any routable subnet in the network, and often isn't to avoid confusion.

These router types should not be confused with the terms *designated router (DR)*, or *backup designated router (BDR)*, which are attributes of a router interface, not the router itself.

Area border router

An area border router (ABR) is a router that connects one or more areas to the main backbone network. It is considered a member of all areas it is connected to. An ABR keeps multiple copies of the link-state database in memory, one for each area to which that router is connected.

Autonomous system boundary router

An autonomous system boundary router (ASBR) is a router that is connected to more than one Routing protocol and that exchanges routing information with routers in other protocols. ASBRs typically also run an exterior routing protocol (e.g., BGP), or use static routes, or both. An ASBR is used to distribute routes received from other, external ASs throughout its own autonomous system. (An interactive ASBR simulation [14] shows how an ASBR creates External LSA for external addresses and floods them to all areas via ABR.) Routers in other areas use ABR as next hop to access external addresses. Then ABR forwards packets to the ASBR that announces the external addresses.

Internal router

An internal router is a router that has OSPF neighbor relationships with interfaces in the same area. An internal router has all its interfaces in a single area.

Backbone router

Backbone routers are all routers that are connected to the OSPF backbone, irrespective of whether they are also area border routers or internal routers of the backbone area. An area border router is always a backbone router, since all areas must be either directly connected to the backbone or connected to the backbone via a virtual link (spanning across another area to get to the backbone).

Designated router

A *designated router* (DR) is the router interface elected among all routers on a particular multiaccess network segment, generally assumed to be broadcast multiaccess. A DR Election Simulation [15] visualizes the basic neighbor discovery process (Hello), flooding (224.0.0.6), DR election (priority, RID). Special techniques, often vendor-dependent, may be needed to support the DR function on nonbroadcast multiaccess (NBMA) media. It is usually wise to configure the individual virtual circuits of a NBMA subnet as individual point-to-point lines; the techniques used are implementation-dependent.

Do not confuse the DR with an OSPF router type. A given physical router can have some interfaces that are designated (DR), others that are backup designated (BDR), and others that are non-designated. If no router is DR or BDR on a given subnet, the DR is first elected, and then a second election is held if there is more than one BDR.[16] (A DR Election Detail Simulation [17] shows a step-by-step DR election example: How neighbor list, neighbor state, DR, and BDR are changed when receiving Hello) The DR is elected based on the following default criteria:

- If the priority setting on an OSPF router is set to 0, that means it can NEVER become a DR or BDR (Backup Designated Router).
- When a DR fails and the BDR takes over, there is another election to see who becomes the replacement BDR.
- The router sending the Hello packets with the highest priority wins the election.
- If two or more routers tie with the highest priority setting, the router sending the Hello with the highest RID (Router ID) wins. NOTE: a RID is the highest logical (loopback) IP address configured on a router, if no logical/loopback IP address is set then the Router uses the highest IP address configured on its active interfaces. (e.g. 192.168.0.1 would be higher than 10.1.1.2).
- Usually the router with the second highest priority number becomes the BDR.
- The priority values range between 0 - 255,[18] with a higher value increasing its chances of becoming DR or BDR.
- IF a HIGHER priority OSPF router comes online AFTER the election has taken place, it will not become DR or BDR until (at least) the DR and BDR fail.
- If the current DR 'goes down' the current BDR becomes the new DR and a new election takes place to find another BDR. If the new DR then 'goes down' and the original DR is now available, still previously chosen BDR will become DR.

DR's exist for the purpose of reducing network traffic by providing a source for routing updates. The DR maintains a complete topology table of the network and sends the updates to the other routers via multicast. All routers in a multi-access network segment will form a slave/master relationship with the DR. They will form adjacencies with the DR and BDR only. Every time a router sends an update, it sends it to the DR and BDR on the multicast address 224.0.0.6. The DR will then send the update out to all other routers in the area, to the multicast address 224.0.0.5. This way all the routers do not have to constantly update each other, and can rather get all their updates from a single source. The use of multicasting further reduces the network load. DRs and BDRs are always setup/elected on OSPF broadcast networks. DR's can also be elected on NBMA (Non-Broadcast Multi-Access) networks such as Frame Relay or ATM. DRs or BDRs are not elected on point-to-point links (such as a point-to-point WAN connection)

because the two routers on either sides of the link must become fully adjacent and the bandwidth between them cannot be further optimized. DR LSDB Synch Simulation [19] shows how DR and non-DR routers evolve from 2-way to full adjacency relationships by exchanging DD, Request, and Update.

Backup designated router

A *backup designated router* (BDR) is a router that becomes the designated router if the current designated router has a problem or fails. The BDR is the OSPF router with second highest priority at the time of the last election.

OSPF v3 Packet Formats

The "Main OSPF Packet Header" is the same for all 5 types of packets (with exception of the Type field) where as the following sub-headers will vary from type to type and are shown below the Main OSPF Packet Header.

The Main OSPF Packet Header

Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version								Type								Packet Length															
4	32	Router ID																															
8	64	Area ID																															
12	96	Checksum																Instance ID								0							

As per Appendix A.3 of RFC 5340 (OSPFv3 for IPv6) there are 5 OSPF Packet formats as follows:

Type	Description
1	Hello
2	Database Description
3	Link State Request
4	Link State Update
5	Link State Acknowledgement

The five different formats for each "Type" of OSPF v3 packet are listed below:

Type 1: The Hello Packet

Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	3 {Ver}								1 {Type}								Packet Length															
4	32	Router ID																															
8	64	Area ID																															
12	96	Checksum																Instance ID								0							

16	**128**	Interface ID	
20	**160**	Rtr Priority	Options (Explained below)
24	**192**	HelloInterval	RouterDeadInterval
28	**224**	Designated Router ID	
32	**256**	Backup Designated Router ID	
36	**288**	Neighbor ID	
~	**~**	...	

Type 2: The Database Description Packet

Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	3 {Ver}								2 {Type}								Packet Length															
4	**32**	Router ID																															
8	**64**	Area ID																															
12	**96**	Checksum																Instance ID								0							
16	**128**	0								Options (Explained below)																							
20	**160**	Interface MTU																0								0	0	0	0	0	I	M	M S
24	**192**	DD sequence number																															

28	**224**	An LSA Header
32	**256**	
36	**288**	
40	**320**	
44	**352**	
~	**~**	...

Type 3: The OSPF Link State Request Packet

Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	3 {Ver}								3 {Type}								Packet Length															
4	**32**	Router ID																															
8	**64**	Area ID																															
12	**96**	Checksum																Instance ID								0							
16	**128**	0																LS Type															
20	**160**	Link State ID																															
24	**192**	Advertising Router																															
~	**~**	...																															

Type 4: The OSPF Link State Update Packet

Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	3 {Ver}								4 {Type}								Packet Length															
4	32	Router ID																															
8	64	Area ID																															
12	96	Checksum																Instance ID								0							
16	128	# LSAs																															
20	160	LSAs																															
24	192																																
28	224																																
32	256																																
36	288																																
~	~	...																															

Type 5: The OSPF Link State Acknowledgement Packet

Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	3 {Ver}								5 {Type}								Packet Length															
4	32	Router ID																															
8	64	Area ID																															
12	96	Checksum																Instance ID								0							

16	**128**	An LSA Header (Shown below)
20	**160**	
24	**192**	
28	**224**	
32	**256**	
~	**~**	...

The OSPFv3 (24 Bit) Options Field

This "Options Field" is used in OSPF Hello packets, Database Description packets, and certain LSAs (router-LSAs, network-LSAs, inter-area-router-LSAs, and link-LSAs).

(Note: Previous OSPF versions {v1 & v2} DO NOT not support all of the options/fields listed here.)

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
																	*	*	DC	R	N	x	E	V6

Explanation of the bits in the Options field:

There are currently only 7-bits assigned.

V6-bit: "V6" stands for IP[v6] routing calculations are to be used.

E-bit: "E" stands for [E]xternal as in AS-External-LSA flooding as specified in OSPFv2.

x-bit: This is currently depreciated. It was previously used by MOSPF.

N-bit: "N" stands for [N]SSA (Not So Stubby Area) and used for routers which are attached to NSSA networks.

R-bit: "R" stands for [R]outer and specifies whether the router is Active or not.

DC-bit: "DC" stands for [D]emand [C]ircuts and is specified in RFC 1793.

*-bits: These two bits are reserved for migration of OSPFv2 protocol extensions.

The remaining 16-bits have yet to be assigned.

OSPF in broadcast and non-broadcast networks

In broadcast multiple-access networks, neighbor adjacency is formed dynamically using multicast hello packets to 224.0.0.5. A DR and BDR are elected normally, and function normally.

For non-broadcast multiple-access networks (NBMA), RFC 2328 defined the following two official modes for OSPF:

- nonbroadcast
- point-to-multipoint

Cisco has defined the following three additional modes for OSPF in NBMA topologies:

- point-to-multipoint nonbroadcast
- broadcast
- point-to-point

Implementations

- BIRD implements both OSPFv2 and OSPFv3
- GNU Zebra, a GPL routing suite for Unix-like systems supporting OSPF
- Netware implements OSPF in its Multi Protocol Routing module.
- OpenBSD includes an OpenOSPFD implementation within the OpenBGPD protocol.
- Quagga, a fork of GNU Zebra for Unix-like systems
- XORP, a routing suite implementing RFC2328 (OSPFv2) and RFC2740 (OSPFv3) for both IPv4 and IPv6
- Windows NT 4.0 Server, Windows 2000 Server and Windows Server 2003 implement OSPFv2 in the Routing and Remote Access Service, although the functionality was removed in Windows Server 2008.

Applications

OSPF was the first widely deployed routing protocol that could converge a network in the low seconds, and guarantee loop-free paths. It has many features that allow the imposition of policies about the propagation of routes that it may be appropriate to keep local, for load sharing, and for selective route importing more than IS-IS. IS-IS, in contrast, can be tuned for lower overhead in a stable network, the sort more common in ISP than enterprise networks. There are some historical accidents that made IS-IS the preferred IGP for ISPs, but ISP's today may well choose to use the features of the now-efficient implementations of OSPF,[20] after first considering the pros and cons of IS-IS in service provider environments.[21]

As mentioned, OSPF can provide better load-sharing on external links than other IGPs. When the default route to an ISP is injected into OSPF from multiple ASBRs as a Type I external route and the same external cost specified, other routers will go to the ASBR with the least path cost from its location. This can be tuned further by adjusting the external cost.

In contrast, if the default route from different ISPs is injected with different external costs, as a Type II external route, the lower-cost default becomes the primary exit and the higher-cost becomes the backup only.

The only real limiting factor that may compel major ISPs to select IS-IS over OSPF is if they have a network with more than 850 routers. There is mention of an OSPF network with over 1000 routers,[22] but that is quite uncommon and the network must be specifically designed to minimize overhead to achieve stable operation.

References

[1] Moy, J. (April 1998). [RFC 2328 "OSPF Version 2"]. The Internet Society. OSPFv2. RFC 2328. Retrieved 2007-09-28.

[2] Coltun, R.; D. Ferguson, J Moy, A. Lindem (July 2008). [RFC 5340 "OSPF for IPv6"]. The Internet Society. OSPFv3. RFC 5340. Retrieved 2008-07-23.

[3] "RFC 2328 - OSPF Version 2" (http://tools.ietf.org/html/rfc2328#page-185). Tools.ietf.org. 1998-04-02. . Retrieved 2011-11-30.

[4] "RFC 5340 - OSPF for IPv6" (http://tools.ietf.org/html/rfc5340#page-57). Tools.ietf.org. . Retrieved 2011-11-30.

[5] RFC 1584, *Multicast Extensions to OSPF*, J. Moy, The Internet Society (March 1994)

[6] http://pre.visualland.net/view.php?cid=915&protocol=OSPF&title=1.5%20Neighbor%20State

[7] http://pre.visualland.net/view.php?cid=931&protocol=OSPF&title=1.6%20Area%20Border%20Router

[8] Hawkinson, J; T. Bates (March 1996). "Guidelines for creation, selection, and registration of an Autonomous System" (http://tools.ietf.org/html/rfc1930). Internet Engineering Task Force. ASguidelines. . Retrieved 2007-09-28.

[9] http://pre.visualland.net/view.php?cid=760&protocol=OSPF&title=3.1%20Stub%20Area

[10] http://pre.visualland.net/view.php?cid=761&protocol=OSPF&title=3.2%20NSSA

[11] "| Under "Stub Area Design Golden Rules" But this is not necessarily true. If there are multiple ABRs, as might be required for high availability, routers interior to the TSA will send non-intra-area traffic to the ABR with the lowest intra-area metric (the "closest" ABR) but that requires special configuration.[http://www.nil.si/ipcorner/OSPFDefaultMysteries/ P.7 "area nssa no-summary" explanation" (http://www.groupstudy.com/bookstore/samples/thomas/). Groupstudy.com. . Retrieved 2011-11-30.

[12] Katz, D; D. Yeung (September 2003). [RFC 3630 "Traffic Engineering (TE) Extensions to OSPF Version 2"]. The Internet Society. OSPF-TEextensions. RFC 3630. Retrieved 2007-09-28.

[13] Rajagopalan, B; J. Luciani & D. Awduche (March 2004). [RFC 3717 "IP over Optical Networks: A Framework"]. Internet Engineering Task Force. OSPFoverOptical. RFC 3717. Retrieved 2007-09-28.

[14] http://pre.visualland.net/view.php?cid=927&protocol=OSPF&title=1.7%20AS%20Border%20Router

[15] http://pre.visualland.net/view.php?cid=924&protocol=OSPF&title=1.2%20DR%20election%20basic

[16] RFC 2328, page 75

[17] http://pre.visualland.net/view.php?cid=940&protocol=OSPF&title=1.3%20DR%20election

[18] "This documentation has been moved. - This documentation has been moved. [Support] - Cisco Systems" (http://www.cisco.com/en/US/docs/ios/iproute/command/reference/irp_osp2.html#wp1012171). Cisco.com. . Retrieved 2011-11-30.

[19] http://pre.visualland.net/view.php?cid=941&protocol=OSPF&title=1.4%20DR%20LSDB%20Synch

[20] Berkowitz, Howard (1999). "OSPF Goodies for ISPs" (http://www.nanog.org/meetings/nanog17/abstracts.php?pt=MTE0OSZuYW5vZzE3&nm=nanog17). *North American Network Operators Group NANOG 17*. Montreal. OSPFforISPs.

[21] Katz, Dave (2000). "OSPF and IS-IS: A Comparative Anatomy" (http://www.nanog.org/meetings/nanog19/abstracts.php?pt=MTA4NCZuYW5vZzE5&nm=nanog19). *North American Network Operators Group NANOG 19*. Albuquerque. OSPFvsISIS.

[22] "CCIE, CCNA, CCNP and other Cisco Certifications" (http://www.groupstudy.com/archives/cisco/200011/msg01985.html). GroupStudy.com. . Retrieved 2011-11-30.

Further reading

- Andrew Colton, *OSPF for Cisco Routers* (http://www.amazon.com/dp/0972286217/)
- Jeff Doyle, Jennifer Carroll, *Routing TCP/IP* (http://www.ciscopress.com/bookstore/product.asp?isbn=1587052024), Volume 1, 2nd Edition
- John T. Moy, *OSPF: Anatomy of an Internet Routing Protocol* (http://www.amazon.com/OSPF-Anatomy-Internet-Routing-Protocol/dp/0201634724/).
- William R. Parkhurst, *Cisco OSPF Command and Configuration Handbook*. ISBN 978-1-58705-071-8
- "Configuring OSPF Authentication" (http://www.netcordia.com/resources/tech-tips/configuring-ospf-authentication.asp). Retrieved 2009-09-10.
- Basu, Anindya; J. Riecke (August 2001). "Stability Issues in OSPF Routing" (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.6393&rep=rep1&type=pdf&rct=j&q=ospf mobility problems&ei=M02STbbpA833gAeJ3Z0Z&usg=AFQjCNHvCTPAOnLe4sM2hpYnlkRRtwjchw&sig2=6i05ABvVeO-y5siqJSnmqw). ACM. 1-58113-411-8/01/0008. Retrieved 2011-03-29.

External links

- IETF OSPF Working Group (http://www.ietf.org/html.charters/ospf-charter.html)
- OSPF Basics (http://www.setup32.com/network-administration/networking/know-ospf.php)
- Cisco OSPF (http://www.cisco.com/en/US/tech/tk365/tk480/tsd_technology_support_sub-protocol_home.html)
- Cisco OSPF Areas and Virtual Links (http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094aaa.shtml)
- Summary of OSPF v2 (http://www.freesoft.org/CIE/Topics/89.htm)
- OSPF and IS-IS: A Comparative Anatomy (http://www.nanog.org/meetings/nanog19/presentations/katz.ppt) by Dave Katz, Juniper
- IS-IS and OSPF difference discussion (http://www.join.uni-muenster.de/Dokumente/drafts/draft-bhatia-manral-diff-isis-ospf-01.txt) (Vishwas Manral, Manav bhatia and Yasuhiro Ohara)
- Data Communication Lectures of Manfred Lindner - Part OSPF Fundamentals (http://www.ict.tuwien.ac.at/lva/384.081/infobase/L41-OSPF_Fundamentals_v4-5.pdf)
- Data Communication Lectures of Manfred Lindner - Part OSPF Areas (http://www.ict.tuwien.ac.at/lva/384.081/infobase/L42-OSPF_Advanced_v4-4.pdf)
- Good comparison of IS-IS vs OSPF (http://www.ecse.rpi.edu/Homepages/koushik/shivkuma-teaching/sp2002/ip2002-isis-ospf.pdf)
- An OSPF Basic Simulation. (http://pre.visualland.net/view.php?cid=767&protocol=OSPF&title=1.1 OSPF basic) The goal is to help beginners grasp the basic ideas quickly. It visualizes the interactions of 4 types of activities on a simplified topology: OSPF tasks (neighbor discovery neighbors, address learning); OSPF packets (Hello, DD, Request, Update); Table update (LSDB, routing table); Ping verification.

Path vector protocol

A **path vector protocol** is a computer network routing protocol which maintains the path information that gets updated dynamically. Updates which have looped through the network and returned to the same node are easily detected and discarded. This algorithm is sometimes used in Bellman–Ford routing algorithms to avoid "Count to Infinity" problems.

It is different from the distance vector routing and link state routing. Each entry in the routing table contains the destination network, the next router and the path to reach the destination.

Path Vector Messages in BGP: The autonomous system boundary routers (ASBR), which participate in path vector routing, advertise the reachability of networks. Each router that receives a path vector message must verify that the advertised path is according to its policy. If the messages comply with the policy, the ASBR modifies its routing table and the message before sending it to the next neighbor. In the modified message it sends its own AS number and replaces the next router entry with its own identification.

BGP is an example of a path vector protocol. In BGP the routing table maintains the autonomous systems that are traversed in order to reach the destination system. *Exterior Gateway Protocol* (EGP) does not use path vectors.

Policy-based routing

In computer networking, **policy-based routing** (**PBR**) is a technique used to make routing decisions based on policies set by the network administrator.

When a router receives a packet it normally decides where to forward it based on the destination address in the packet, which is then used to look up an entry in a routing table. However, in some cases, there may be a need to forward the packet based on other criteria. For example, a network administrator might want to forward a packet based on the source address, not the destination address. This should not be confused with source routing.

Policy-based routing may also be based on the size of the packet, the protocol of the payload, or other information available in a packet header or payload. This permits routing of packets originating from different sources to different networks even when the destinations are the same and can be useful when interconnecting several private networks.

In the Cisco IOS, PBR is implemented using *route maps*.[1] .

Linux supports multiple routing tables since version 2.2.[2]

FreeBSD supports PBR using either IPFW or OpenBSD's PF.

Examples of PBR Usage

PBR can be used to redirect traffic to a proxy server by using a PBR-aware L3-switch (router). In such deployment, specific source traffic (e.g. HTTP, FTP) can be redirected to a cache engine. This is known as virtual inline deployment.

References

[1] Cisco IOS Release 12.0 Quality of Service Solutions Configuration Guide (http://www.cisco.com/en/US/docs/ios/12_0/qos/configuration/guide/qcpolicy.html)

[2] Linux Advanced Routing & Traffic Control HOWTO (http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/index.html)

External links

- Policy routing Cisco Press article (http://www.ciscopress.com/content/downloads/cisco/bookreg/2237xxd.pdf)
- Policy based routing with Linux (ONLINE edition) (http://www.policyrouting.org/PolicyRoutingBook/ONLINE/TOC.html)

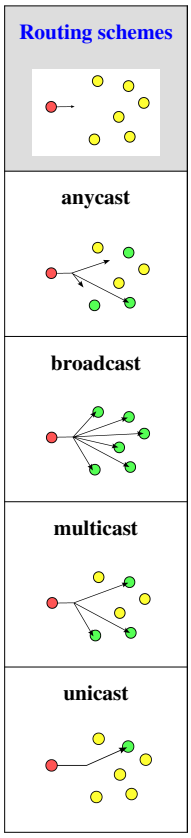
Routing

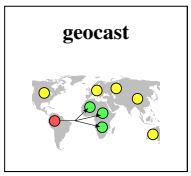
Routing is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (Circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used within localized environments.

Delivery semantics





Routing schemes differ in their delivery semantics:

- unicast delivers a message to a single specific node;
- broadcast delivers a message to all nodes in the network;
- multicast delivers a message to a group of nodes that have expressed interest in receiving the message;
- anycast delivers a message to any one out of a group of nodes, typically the one nearest to the source.
- geocast delivers a message to a geographic area

Unicast is the dominant form of message delivery on the Internet, and this article focuses on unicast routing algorithms.

Topology distribution

In a practice known as static routing (or non-adaptive routing), small networks may use manually configured routing tables. Larger networks have complex topologies that can change rapidly, making the manual construction of routing tables unfeasible. Nevertheless, most of the public switched telephone network (PSTN) uses pre-computed routing tables, with fallback routes if the most direct route becomes blocked (see routing in the PSTN). Adaptive routing, or dynamic routing, attempts to solve this problem by constructing routing tables automatically, based on information carried by routing protocols, and allowing the network to act nearly autonomously in avoiding network failures and blockages.

Examples of adaptive-routing algorithms are the Routing Information Protocol (RIP) and the Open-Shortest-Path-First protocol (OSPF). Adaptive routing dominates the Internet. However, the configuration of the routing protocols often requires a skilled touch; networking technology has not developed to the point of the complete automation of routing.

Distance vector algorithms

Distance vector algorithms use the Bellman-Ford algorithm. This approach assigns a number, the *cost*, to each of the links between each node in the network. Nodes will send information from point A to point B via the path that results in the lowest *total cost* (i.e. the sum of the costs of the links between the nodes used).

The algorithm operates in a very simple manner. When a node first starts, it only knows of its immediate neighbours, and the direct cost involved in reaching them. (This information, the list of destinations, the total cost to each, and the *next hop* to send data to get there, makes up the routing table, or *distance table*.) Each node, on a regular basis, sends to each neighbour its own current idea of the total cost to get to all the destinations it knows of. The neighbouring node(s) examine this information, and compare it to what they already 'know'; anything which represents an improvement on what they already have, they insert in their own routing table(s). Over time, all the nodes in the network will discover the best next hop for all destinations, and the best total cost.

When one of the nodes involved goes down, those nodes which used it as their next hop for certain destinations discard those entries, and create new routing-table information. They then pass this information to all adjacent nodes, which then repeat the process. Eventually all the nodes in the network receive the updated information, and will then discover new paths to all the destinations which they can still "reach".

Link-state algorithms

When applying link-state algorithms, each node uses as its fundamental data a map of the network in the form of a graph. To produce this, each node floods the entire network with information about what other nodes it can connect to, and each node then independently assembles this information into a map. Using this map, each router then independently determines the least-cost path from itself to every other node using a standard shortest paths algorithm such as Dijkstra's algorithm. The result is a tree rooted at the current node such that the path through the tree from the root to any other node is the least-cost path to that node. This tree then serves to construct the routing table, which specifies the best next hop to get from the current node to any other node.

Optimised Link State Routing algorithm

A link-state routing algorithm optimised for mobile ad-hoc networks is the *Optimised Link State Routing Protocol (OLSR)*.[1] OLSR is proactive; it uses Hello and Topology Control (TC) messages to discover and disseminate link state information through the mobile ad-hoc network. Using Hello messages, each node discovers 2-hop neighbor information and elects a set of *multipoint relays* (MPRs). MPRs distinguish OLSR from other link state routing protocols.

Path vector protocol

Distance vector and link state routing are both intra-domain routing protocols. They are used inside an autonomous system, but not between autonomous systems. Both of these routing protocols become intractable in large networks and cannot be used in Inter-domain routing. Distance vector routing is subject to instability if there are more than a few hops in the domain. Link state routing needs huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding.

Path vector routing is used for inter-domain routing. It is similar to distance vector routing. In path vector routing we assume there is one node (there can be many) in each autonomous system which acts on behalf of the entire autonomous system. This node is called the speaker node. The speaker node creates a routing table and advertises it to neighboring speaker nodes in neighboring autonomous systems. The idea is the same as distance vector routing except that only speaker nodes in each autonomous system can communicate with each other. The speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems. Path vector routing is discussed in RFC 1322; the path vector routing algorithm is somewhat similar to the distance vector algorithm in the sense that each border router advertises the destinations it can reach to its neighboring router. However, instead of advertising networks in terms of a destination and the distance to that destination, networks are advertised as destination addresses and path descriptions to reach those destinations. A route is defined as a pairing between a destination and the attributes of the path to that destination, thus the name, path vector routing, where the routers receive a vector that contains paths to a set of destinations. The path, expressed in terms of the domains (or confederations) traversed so far, is carried in a special path attribute that records the sequence of routing domains through which the reachability information has passed.

Comparison of routing algorithms

Distance-vector routing protocols are simple and efficient in small networks and require little, if any, management. However, traditional distance-vector algorithms have poor convergence properties due to the count-to-infinity problem.

This has led to the development of more complex but more scalable algorithms for use in large networks. Interior routing mostly uses link-state routing protocols such as OSPF and IS-IS.

A more recent development is that of loop-free distance-vector protocols (e.g., EIGRP). Loop-free distance-vector protocols are as robust and manageable as naive distance-vector protocols, but avoid counting to infinity, and have good worst-case convergence times.

Path selection

Path selection involves applying a routing metric to multiple routes, in order to select (or predict) the best route.

In the case of computer networking, the metric is computed by a routing algorithm, and can cover such information as bandwidth, network delay, hop count, path cost, load, MTU, reliability, and communication cost (see e.g. this survey [2] for a list of proposed routing metrics). The routing table stores only the best possible routes, while link-state or topological databases may store all other information as well.

Because a routing metric is specific to a given routing protocol, multi-protocol routers must use some external heuristic in order to select between routes learned from different routing protocols. Cisco's routers, for example, attribute a value known as the administrative distance to each route, where smaller administrative distances indicate routes learned from a supposedly more reliable protocol.

A local network administrator, in special cases, can set up host-specific routes to a particular machine which provides more control over network usage, permits testing and better overall security. This can come in handy when required to debug network connections or routing tables.

Multiple agents

In some networks, routing is complicated by the fact that no single entity is responsible for selecting paths: instead, multiple entities are involved in selecting paths or even parts of a single path. Complications or inefficiency can result if these entities choose paths to optimize their own objectives, which may conflict with the objectives of other participants.

A classic example involves traffic in a road system, in which each driver picks a path which minimizes their own travel time. With such routing, the equilibrium routes can be longer than optimal for all drivers. In particular, Braess paradox shows that adding a new road can *lengthen* travel times for all drivers.

In another model, for example used for routing automated guided vehicles (AGVs) on a terminal, reservations are made for each vehicle to prevent simultaneous use of the same part of an infrastructure. This approach is also referred to as context-aware routing.[2]

The Internet is partitioned into autonomous systems (ASs) such as internet service providers (ISPs), each of which has control over routes involving its network, at multiple levels. First, AS-level paths are selected via the BGP protocol, which produces a sequence of ASs through which packets will flow. Each AS may have multiple paths, offered by neighboring ASs, from which to choose. Its decision often involves business relationships with these neighboring ASs,[3] which may be unrelated to path quality or latency. Second, once an AS-level path has been selected, there are often multiple corresponding router-level paths, in part because two ISPs may be connected in multiple locations. In choosing the single router-level path, it is common practice for each ISP to employ hot-potato routing: sending traffic along the path that minimizes the distance through the ISP's own network—even if that path lengthens the total distance to the destination.

Consider two ISPs, *A* and *B*, which each have a presence in New York, connected by a fast link with latency 5 ms; and which each have a presence in London connected by a 5 ms link. Suppose both ISPs have trans-Atlantic links connecting their two networks, but *A*'s link has latency 100 ms and B's has latency 120 ms. When routing a message from a source in *A*'s London network to a destination in *B*'s New York network, *A* may choose to immediately send the message to *B* in London. This saves *A* the work of sending it along an expensive trans-Atlantic link, but causes the message to experience latency 125 ms when the other route would have been 20 ms faster.

A 2003 measurement study of Internet routes found that, between pairs of neighboring ISPs, more than 30% of paths have inflated latency due to hot-potato routing, with 5% of paths being delayed by at least 12 ms. Inflation due to AS-level path selection, while substantial, was attributed primarily to BGP's lack of a mechanism to directly optimize for latency, rather than to selfish routing policies. It was also suggested that, were an appropriate mechanism in place, ISPs would be willing to cooperate to reduce latency rather than use hot-potato routing.[4]

Such a mechanism was later published by the same authors, first for the case of two ISPs[5] and then for the global case.[6]

Route Analytics

As the Internet and IP networks become mission critical business tools, there has been increased interest in techniques and methods to monitor the routing posture of networks. Incorrect routing or routing issues cause undesirable performance degradation, flapping and/or downtime. Monitoring routing in a network is achieved using Route analytics tools and techniques

References

[1] RFC 3626

[2] Jonne Zutt, Arjan J.C. van Gemund, Mathijs M. de Weerdt, and Cees Witteveen (2010). Dealing with Uncertainty in Operational Transport Planning (http://www.st.ewi.tudelft.nl/~mathijs/publications/intinfra09.pdf). In R.R. Negenborn and Z. Lukszo and H. Hellendoorn (Eds.) Intelligent Infrastructures, Ch. 14, pp. 355-382. Springer.

[3] Matthew Caesar and Jennifer Rexford. BGP routing policies in ISP networks (http://www.cs.princeton.edu/~jrex/papers/policies.pdf). IEEE Network Magazine, special issue on Interdomain Routing, Nov/Dec 2005.

[4] Neil Spring, Ratul Mahajan, and Thomas Anderson. Quantifying the Causes of Path Inflation (http://www.cs.washington.edu/research/networking/rocketfuel/papers/sigcomm2003.pdf). Proc. SIGCOMM 2003.

[5] Ratul Mahajan, David Wetherall, and Thomas Anderson. Negotiation-Based Routing Between Neighboring ISPs (http://research.microsoft.com/en-us/um/people/ratul/papers/nsdi2005-nexit.pdf). Proc. NSDI 2005.

[6] Ratul Mahajan, David Wetherall, and Thomas Anderson. Mutually Controlled Routing with Independent ISPs (http://research.microsoft.com/en-us/um/people/ratul/papers/nsdi2007-wiser.pdf). Proc. NSDI 2007.

- Ash, Gerald (1997). *Dynamic Routing in Telecommunication Networks*. McGraw-Hill. ISBN 0070064148.
- Doyle, Jeff and Carroll, Jennifer (2005). *Routing TCP/IP, Volume I, Second Ed.*. Cisco Press. ISBN 1587052024. Ciscopress ISBN 1-58705-202-4 (http://www.ciscopress.com/title/1587052024)
- Doyle, Jeff and Carroll, Jennifer (2001). *Routing TCP/IP, Volume II,*. Cisco Press. ISBN 1578700892. Ciscopress ISBN 1-57870-089-2 (http://www.ciscopress.com/title/1578700892)
- Huitema, Christian (2000). *Routing in the Internet, Second Ed.*. Prentice-Hall. ISBN 0321227352.
- Kurose, James E. and Ross, Keith W. (2004). *Computer Networking, Third Ed.*. Benjamin/Cummings. ISBN 0321227352.
- Medhi, Deepankar and Ramasamy, Karthikeyan (2007). *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann. ISBN 0120885883.

External links

- Count-To-Infinity Problem (http://wiki.uni.lu/secan-lab/Count-To-Infinity+Problem.html)
- "Stability Features" (http://wwwlehre.dhbw-stuttgart.de/~schulte/htme/55024.htm#HDR3) are ways of avoiding the "count to infinity" problem.
- Cisco IT Case Studies (http://www.cisco.com/web/about/ciscoitatwork/case_studies/routing.html) about Routing and Switching

Routing Information Protocol

The **Routing Information Protocol** (**RIP**) is a distance-vector routing protocol, which employs the hop count as a routing metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops allowed for RIP is 15. This hop limit, however, also limits the size of networks that RIP can support. A hop count of 16 is considered an infinite distance and used to deprecate inaccessible, inoperable, or otherwise undesirable routes in the selection process.

RIP implements the split horizon, route poisoning and holddown mechanisms to prevent incorrect routing information from being propagated. These are some of the stability features of RIP. It is also possible to use the so called RMTI[1] (**R**outing Information Protocol with **M**etric-based **T**opology **I**nvestigation) algorithm to cope with the count-to-infinity problem. With its help, it is possible to detect every possible loop with a very small computation effort.

Originally each RIP router transmitted full updates every 30 seconds. In the early deployments, routing tables were small enough that the traffic was not significant. As networks grew in size, however, it became evident there could be a massive traffic burst every 30 seconds, even if the routers had been initialized at random times. It was thought, as a result of random initialization, the routing updates would spread out in time, but this was not true in practice. Sally Floyd and Van Jacobson showed in 1994[2] that, without slight randomization of the update timer, the timers synchronized over time. In most current networking environments, RIP is not the preferred choice for routing as its time to converge and scalability are poor compared to EIGRP, OSPF, or IS-IS (the latter two being link-state routing protocols), and (without RMTI) a hop limit severely limits the size of network it can be used in. However, it is easy to configure, because RIP does not require any parameters on a router unlike other protocols (see here [3] for an animation of basic RIP simulation visualizing RIP configuration and exchanging of Request and Response to discover new routes).

RIP is implemented on top of the User Datagram Protocol as its transport protocol. It is assigned the reserved port number 520.[4]

Versions

There are three versions of the Routing Information Protocol: *RIPv1*, *RIPv2*, and *RIPng*.

RIP version 1

The original specification of RIP, defined in RFC 1058,[5] uses classful routing. The periodic routing updates do not carry subnet information, lacking support for variable length subnet masks (VLSM). This limitation makes it impossible to have different-sized subnets inside of the same network class. In other words, all subnets in a network class must have the same size. There is also no support for router authentication, making RIP vulnerable to various attacks.

RIP version 2

Due to the deficiencies of the original RIP specification, RIP version 2 (RIPv2) was developed in 1993[6] and last standardized in 1998.[7] It included the ability to carry subnet information, thus supporting Classless Inter-Domain Routing (CIDR). To maintain backward compatibility, the hop count limit of 15 remained. RIPv2 has facilities to fully interoperate with the earlier specification if all *Must Be Zero* protocol fields in the RIPv1 messages are properly specified. In addition, a *compatibility switch* feature[7] allows fine-grained interoperability adjustments.

In an effort to avoid unnecessary load on hosts that do not participate in routing, RIPv2 *multicasts* the entire routing table to all adjacent routers at the address 224.0.0.9, as opposed to RIPv1 which uses broadcast. Unicast addressing is still allowed for special applications.

(MD5) authentication for RIP was introduced in 1997.[8] [9]

RIPv2 is Internet Standard STD56 (which is RFC 2453).

Route tags were also added in RIP version 2. This functionality allows for routes to be distinguished from internal routes to external redistributed routes from EGP protocols.

RIPng

RIPng (RIP next generation), defined in RFC 2080,[10] is an extension of RIPv2 for support of IPv6, the next generation Internet Protocol. The main differences between RIPv2 and RIPng are:

- Support of IPv6 networking.
- While RIPv2 supports RIPv1 updates authentication, RIPng does not. IPv6 routers were, at the time, supposed to use IPsec for authentication.
- RIPv2 allows attaching arbitrary tags to routes, RIPng does not;
- RIPv2 encodes the next-hop into each route entries, RIPng requires specific encoding of the next hop for a set of route entries.

Limitations

- Without using RMTI, Hop count can not exceed 15, in the case that it exceeds this limitation, it will be considered invalid.
- Most RIP networks are flat. There is no concept of areas or boundaries in RIP networks.
- Variable Length Subnet Masks were not supported by RIP version 1.
- Without using RMTI, RIP has slow convergence and count to infinity problems.

Implementations

- Cisco IOS, software used in Cisco routers (supports version 1, version 2 and RIPng)
- Cisco NX-OS software used in Cisco Nexus data center switches (supports RIPv1 and RIPv2)
- Junos software used in Juniper routers, switches, and firewalls (supports RIPv1 and RIPv2)
- routed[11] , included in most BSD Unix systems
- Routing and Remote Access, a Windows Server feature, contains RIP support
- Quagga, a free open source routing software suite based on GNU Zebra
- BIRD, a free open source routing software suite
- OpenBSD, includes a RIP implementation

Similar protocols

- **IGRP**: Cisco's proprietary Interior Gateway Routing Protocol (IGRP) was a somewhat more capable protocol than RIP. It belongs to the same basic family of distance-vector routing protocols. Cisco has ceased support and distribution of IGRP in their router software. It was replaced by the Enhanced Interior Gateway Routing Protocol (EIGRP) which is a completely new design. While EIGRP still uses a distance-vector model, it relates to IGRP only in using the same routing metrics. IGRP supports multiple metrics for each route, including bandwidth, delay, load, MTU, and reliability.

References

[1] "RMTI project" (http://userp.uni-koblenz.de/~vnuml/rmti/). .
[2] The Synchronization of Periodic Routing Messages (http://www.icir.org/floyd/papers/sync_94.pdf), S. Floyd & V. Jacobson,April 1994
[3] http://www.learningocean.com/view.php?cid=899&protocol=RIP&title=1.%20RIP%20Basic&ctype=1
[4] "*Port Numbers*" (http://www.iana.org/assignments/port-numbers) (plain text). The Internet Assigned Numbers Authority (IANA). 2008-05-22. . Retrieved 2008-05-25.
[5] RFC 1058, *Routing Information Protocol*, C. Hendrik, The Internet Society (June 1988)
[6] RFC 1388, *RIP Version 2 - Carrying Additional Information*, G. Malkin, The Internet Society (January 1993)
[7] RFC 2453, *RIP Version 2*, G. Malkin, The Internet Society (November 1998)
[8] RFC 2082, *RIP-2 MD5 Authentication*, F. Baker, R. Atkinson, The Internet Society (January 1997)
[9] RFC 4822, *RIPv2 Cryptographic Authentication*, R. Atkinson, M. Fanto, The Internet Society (January 2007)
[10] RFC 2080, *RIPng for IPv6*, G. Malkin, R. Minnear, The Internet Society (January 1997)
[11] Man-page for "routed(8)" on FreeBSD, http://www.freebsd.org/cgi/man.cgi?query=routed&sektion=8

Further reading

- Malkin, Gary Scott (2000). *RIP: An Intra-Domain Routing Protocol*. Addison-Wesley Longman. ISBN 0-201-43320-6.
- Edward A. Taft, *Gateway Information Protocol (revised)* (Xerox Parc, Palo Alto, May, 1979)
- *Xerox System Integration Standard - Internet Transport Protocols* (Xerox, Stamford, 1981)

External links

- Interactive RIP simulation (http://www.visualland.net/view.php?cid=899&protocol=RIP&title=1.RIP Basic&ctype=1)

Routing protocol

A **routing protocol** is a protocol that specifies how routers communicate with each other, disseminating information that enables them to select routes between any two nodes on a computer network, the choice of the route being done by routing algorithms. Each router has *a priori* knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors, and then throughout the network. This way, routers gain knowledge of the topology of the network. For a discussion of the concepts behind routing protocols, see: Routing.

The term **routing protocol** may refer specifically to one operating at layer three of the OSI model, which similarly disseminates topology information between routers.

Although there are many types of routing protocols, three major classes are in widespread use on IP networks:

- Interior gateway routing via link-state routing protocols, such as OSPF and IS-IS
- Interior gateway routing via path vector or distance vector protocols, such as RIP, IGRP and EIGRP
- Exterior gateway routing. BGP v4 is the routing protocol used by the public Internet.

Many routing protocols are defined in documents called RFCs.[1] [2] [3] [4]

The specific characteristics of routing protocols include

- the manner in which they either prevent routing loops from forming or break them up if they do
- the manner in which they select preferred routes, using information about hop costs
- the time they take to converge
- how well they scale up
- many other factors

Routed versus routing protocols

A *routed* protocol can be routed by a router, i.e., it can be forwarded from one router to another. A *routing* protocol sends and receives packets containing routing information to and from other routers.

In some cases, routing protocols can themselves run over routed protocols: for example, BGP runs over TCP which runs over IP; care is taken in the implementation of such systems not to create a circular dependency between the routing and routed protocols. That a routing protocol runs over particular transport mechanism does not mean that the routing protocol is of layer (N+1) if the transport mechanism is of layer (N). Routing protocols, according to the OSI Routing framework, are layer management protocols for the network layer, regardless of their transport mechanism:

- IS-IS runs over the data link layer
- OSPF, IGRP, and EIGRP run directly over IP; OSPF and EIGRP have their own reliable transmission mechanism while IGRP assumed an unreliable transport
- RIP runs over UDP
- BGP runs over TCP

Examples

Interior routing protocols

Interior Gateway Protocols (IGPs) exchange routing information within a single routing domain. A given autonomous system [5] can contain multiple routing domains, or a set of routing domains can be coordinated without being an Internet-participating autonomous system. Common examples include:

- IGRP (Interior Gateway Routing Protocol)
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- OSPF (Open Shortest Path First)
- RIP (Routing Information Protocol)
- IS-IS (Intermediate System to Intermediate System)

Note that IGRP, a Cisco proprietary routing protocol, is no longer supported. EIGRP accepts IGRP configuration commands, but the internals of IGRP and EIGRP are completely different.

References

[1] INTERNET PROTOCOL (ftp://ftp.rfc-editor.org/in-notes/rfc791.txt), RFC 791, J Postel, September 1981.
[2] BROADCASTING INTERNET DATAGRAMS IN THE PRESENCE OF SUBNETS (ftp://ftp.rfc-editor.org/in-notes/rfc922.txt), RFC 922, Jeffrey Mogul, October 1984
[3] Towards Requirements for IP Routers (ftp://ftp.rfc-editor.org/in-notes/rfc1716.txt), RFC 1716, P. Almquist, November 1994
[4] Requirements for IP Version 4 Routers (ftp://ftp.rfc-editor.org/in-notes/rfc1812.txt), RFC 1812, F. Baker,June 1995
[5] Guidelines for creation, selection, and registration of an Autonomous System (AS) (ftp://ftp.rfc-editor.org/in-notes/rfc1930.txt), RFC 1930, J. Hawkison & T. Bates,March1996

Further reading

- Chapter "Routing Basics" (http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook) in the Cisco "Internetworking Technology Handbook"

Routing table

In computer networking a **routing table**, or **Routing Information Base (RIB)**, is a data table stored in a router or a networked computer that lists the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes. The routing table contains information about the topology of the network immediately around it. The construction of routing tables is the primary goal of routing protocols. Static routes are entries made in a routing table by non-automatic means and which are fixed rather than being the result of some network topology 'discovery' procedure.

Basics

A routing table utilizes the same idea that one does when using a map in package delivery. Whenever a node needs to send data to another node on a network, it must know *where* to send it, first. If the node cannot directly connect to the destination node, it has to send it via other nodes along a proper route to the destination node. Most nodes do not try to figure out which route(s) might work; instead, a node will send an IP packet to a gateway in the LAN, which then decides how to route the "package" of data to the correct destination. Each gateway will need to keep track of which way to deliver various packages of data, and for this it uses a Routing Table. A routing table is a database which keeps track of paths, like a map, and allows the gateway to provide this information to the node requesting the information.

With hop-by-hop routing, each routing table lists, for all reachable destinations, the address of the next device along the path to that destination: the *next hop*. Assuming that the routing tables are consistent, the simple algorithm of relaying packets to their destination's next hop thus suffices to deliver data anywhere in a network. Hop-by-hop is the fundamental characteristic of the IP Internetwork Layer [1] and the OSI Network Layer.

Difficulties with routing tables

The need to record routes to large numbers of devices using limited storage space represents a major challenge in routing table construction. In the Internet, the currently dominant address aggregation technology is a bitwise prefix matching scheme called Classless Inter-Domain Routing (CIDR).

Since in a network each node presumably possesses a valid routing table, routing tables must be consistent among the various nodes or routing loops can develop. This is particularly problematic in the hop-by-hop routing model in which the net effect of inconsistent tables in several different routers could be to forward packets in an endless loop. Routing loops have historically plagued routing, and their avoidance is a major design goal of routing protocols.

Contents of routing tables

The routing table consists of at least three information fields:

1. *the network id*: i.e. the destination network id
2. *cost*: i.e. the cost or metric of the path through which the packet is to be sent
3. *next hop*: The next hop, or gateway, is the address of the next station to which the packet is to be sent on the way to its final destination

Depending on the application and implementation, it can also contain additional values that refine path selection:

1. *quality of service* associated with the route. For example, the U flag indicates that an IP route is up.
2. *links to filtering criteria*/access lists associated with the route
3. *interface*: such as eth0 for the first Ethernet card, eth1 for the second Ethernet card, etc.

Routing tables are also a key aspect of certain security operations, such as *unicast reverse path forwarding* (uRPF).[2] In this technique, which has several variants, the router also looks up, in the routing table, the *source*

address of the packet. If there exists no route back to the source address, the packet is assumed to be malformed or involved in a network attack, and is dropped.

Network id	Cost	Next hop
.....	.....	.....
.....	.....	.....

Shown below is an example of what the table above could look like on an average computer connected to the internet via a home router:

Network Destination	Netmask	Gateway	Interface	Metric
0.0.0.0	0.0.0.0	192.168.0.1	192.168.0.100	10
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.0.0	255.255.255.0	192.168.0.100	192.168.0.100	10
192.168.0.100	255.255.255.255	127.0.0.1	127.0.0.1	10
192.168.0.255	255.255.255.255	192.168.0.100	192.168.0.100	10

- The columns **Network Destination** and **Netmask** together describe the **Network id** as mentioned earlier. For example, destination **192.168.0.0** and netmask **255.255.255.0** can be written as network id **192.168.0.0/24**.
- The **Gateway** column contains the same information as the **Next hop**, i. e. it points to the gateway through which the network can be reached.
- The **Interface** indicates what locally available interface is responsible for reaching the gateway. In this example, gateway **192.168.0.1** (the internet router) can be reached through the local network card with address **192.168.0.100**.
- Finally, the **Metric** indicates the associated cost of using the indicated route. This is useful for determining the efficiency of a certain route from two points in a network. In this example, it is more efficient to communicate with the computer itself through the use of address **127.0.0.1** (called “localhost”) than it would be through **192.168.0.100** (the IP address of the local network card).

Forwarding table

Routing tables are generally not used directly for packet forwarding in modern router architectures; instead, they are used to generate the information for a smaller forwarding table. A forwarding table contains only the routes which are chosen by the routing algorithm as preferred routes for packet forwarding. It is often in a compressed or pre-compiled format that is optimized for hardware storage and lookup.

This router architecture separates the Control Plane function of the routing table from the Forwarding Plane function of the forwarding table. [3] This separation of control and forwarding provides uninterrupted forwarding in production environments, along with enhanced performance.

References

[1] Requirements for IPv4 Routers (http://www.ietf.org/rfc/rfc1812.txt), F. Baker, RFC 1812, June 1995
[2] Ingress Filtering for Multihomed Networks (ftp://ftp.rfc-editor.org/in-notes/rfc3704.txt),RFC 3704, F. Baker & P. Savola,March 2004
[3] Forwarding and Control Element Separation (ForCES) Framework (http://www.ietf.org/rfc/rfc3746.txt), L. Yang *et al.*, RFC3746,April 2004.

External links

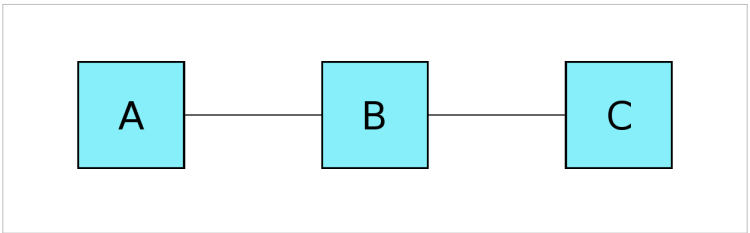
- IP Routing (http://www.faqs.org/docs/linux_network/x-087-2-issues.routing.html)
- Good explanation of the Network destination, Netmask, Gateway, Interface, Metric, etc. (http://technet.microsoft.com/en-us/library/cc779122(WS.10).aspx)
- Static Routing Simulation (http://www.osischool.com/protocol/routing/static-routing)

Split horizon route advertisement

In computer networking, **split-horizon route advertisement** is a method of preventing routing loops in distance-vector routing protocols by prohibiting a router from advertising a route back onto the interface from which it was learned.

Example

In this example, network node *A* routes packets to node *B* in order to reach node *C*. The links between the nodes are distinct point-to-point links.



According to the split-horizon rule, node *A* does not advertise its route for *C* (namely *A* to *B* to *C*) back to *B*. On the surface, this seems redundant since *B* will never route via node *A* because the route costs more than the direct route from *B* to *C*. However, if the link between *B* and *C* goes down, and *B* had received a route from *A*, *B* could end up using that route via *A*. *A* would send the packet right back to *B*, creating a loop. With the split-horizon rule in place, this particular loop scenario cannot happen, improving convergence time in complex, highly-redundant environments.

There are scenarios where the split-horizon rule may fail. Suppose 6 nodes(A,B,C,D,E,F) are each connected to every other node. The lowest cost routes will establish over time. Now if one of the nodes fails. Split-horizon only protects each node from where their current path is received from. But the same information received from the other nodes will be transmitted as well.

If A connects to E via B and C. And if D connects to E via A->B->C. Suppose B-C fail. A will not advertise its route to E to B directly due to the split-horizon rule. However, as each node is connected to each other. D will advertise the route to E to B (D->A->B->C-E). The split horizon rule stops D from advertising the route to A. Not to B.

Such scenarios and multiple links exist in practical networks. And it is the network administrator who needs to work out such intricacies.

Poison reverse

Split-horizon routing with poison reverse[1] is a variant of split-horizon route advertising in which a router actively advertises routes as unreachable over the interface over which they were learned by setting the route metric to infinite (∞). The effect of such an announcement is to immediately remove most looping routes before they can propagate through the network.

The main disadvantage of poison reverse is that it can significantly increase the size of routing announcements in certain fairly common network topologies, but it allows to improve the overall efficiency of the network in case of faults.

Implementations

The split-horizon method is effective and simple to implement, and is therefore used by most distance-vector protocols. It is notably used by:

- RIP
- IGRP
- EIGRP
- VPLS uses the split horizon technique to avoid loops in the forwarding plane.
- Babel applies split-horizon to wired links only.

References

[1] IP routing protocols By Uyless D. Black (http://books.google.com/books?id=CN5MWZddE0QC&lpg=PP1&dq=Distance-vector routing decnet&pg=PA114#v=onepage&q=&f=false)

External links

- "Split Horizon" and "Split Horizon with Poison Reverse" in Convergence in RIP Internetworks (http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/intwork/inae_ips_vzbs.mspx?mfr=true), from Microsoft TechNet.

Application Layer

Domain Name System

The **Domain Name System** (**DNS**) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most importantly, it translates domain names meaningful to humans into the numerical identifiers associated with networking equipment for the purpose of locating and addressing these devices worldwide.

An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. For example, the domain name www.example.com translates to the addresses 192.0.32.10 (IPv4) and 2620:0:2d0:200::10 (IPv6).

The Domain Name System makes it possible to assign domain names to groups of Internet resources and users in a meaningful way, independent of each entity's physical location. Because of this, World Wide Web (WWW) hyperlinks and Internet contact information can remain consistent and constant even if the current Internet routing arrangements change or the participant uses a mobile device. Internet domain names are easier to remember than IP addresses such as `208.77.188.166` (IPv4) or `2001:db8:1f70::999:de8:7648:6e8` (IPv6). Users take advantage of this when they recite meaningful Uniform Resource Locators (URLs) and e-mail addresses without having to know how the computer actually locates them.

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. This mechanism has made the DNS distributed and fault tolerant and has helped avoid the need for a single central register to be continually consulted and updated.

In general, the Domain Name System also stores other types of information, such as the list of mail servers that accept email for a given Internet domain. By providing a worldwide, distributed keyword-based redirection service, the Domain Name System is an essential component of the functionality of the Internet.

Other identifiers such as RFID tags, UPCs, international characters in email addresses and host names, and a variety of other identifiers could all potentially use DNS.[1] [2]

The Domain Name System also specifies the technical functionality of this database service. It defines the DNS protocol, a detailed specification of the data structures and communication exchanges used in DNS, as part of the Internet Protocol Suite.

Overview

The Internet maintains two principal namespaces, the domain name hierarchy[3] and the Internet Protocol (IP) address spaces.[4] The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System.[5] A DNS name server is a server that stores the DNS records for a domain name, such as address (A) records, name server (NS) records, and mail exchanger (MX) records (see also list of DNS record types); a DNS name server responds with answers to queries against its database.

History

The practice of using a name as a simpler, more memorable abstraction of a host's numerical address on a network dates back to the ARPANET era. Before the DNS was invented in 1982, each computer on the network retrieved a file called *HOSTS.TXT* from a computer at SRI (now SRI International).[6] [7] The HOSTS.TXT file mapped names to numerical addresses. A hosts file still exists on most modern operating systems by default and generally contains a mapping of the IP address 127.0.0.1 to "localhost". Many operating systems use name resolution logic that allows the administrator to configure selection priorities for available name resolution methods.

The rapid growth of the network made a centrally maintained, hand-crafted HOSTS.TXT file unsustainable; it became necessary to implement a more scalable system capable of automatically disseminating the requisite information.

At the request of Jon Postel, Paul Mockapetris invented the Domain Name System in 1983 and wrote the first implementation. The original specifications were published by the Internet Engineering Task Force in RFC 882 and RFC 883, which were superseded in November 1987 by RFC 1034[3] and RFC 1035.[5] Several additional Request for Comments have proposed various extensions to the core DNS protocols.

In 1984, four Berkeley students—Douglas Terry, Mark Painter, David Riggle, and Songnian Zhou—wrote the first Unix implementation, called The Berkeley Internet Name Domain (BIND) Server.[8] In 1985, Kevin Dunlap of DEC significantly re-wrote the DNS implementation. Mike Karels, Phil Almquist, and Paul Vixie have maintained BIND since then. BIND was ported to the Windows NT platform in the early 1990s.

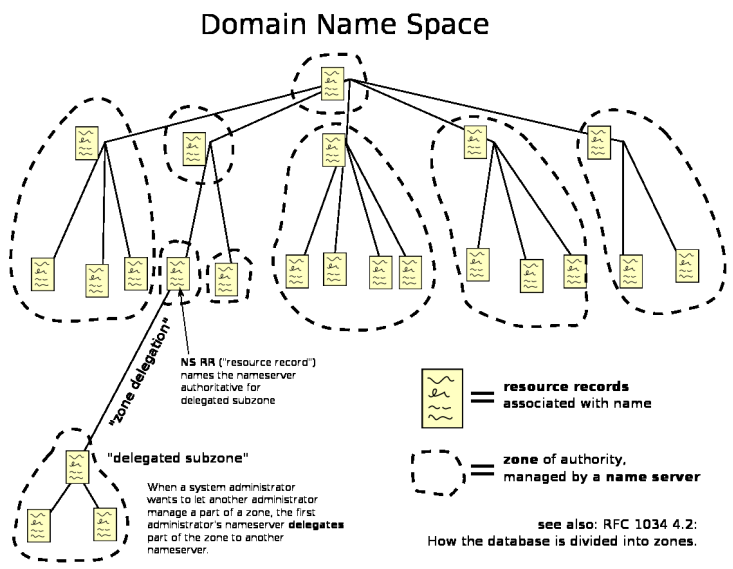
BIND was widely distributed, especially on Unix systems, and is the dominant DNS software in use on the Internet.[9] With the heavy use and resulting scrutiny of its open-source code, as well as increasingly more sophisticated attack methods, many security flaws were discovered in BIND. This contributed to the development of a number of alternative name server and resolver programs. BIND version 9 was written from scratch and now has a security record comparable to other modern DNS software.

Structure

Domain name space

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more *resource records*, which hold information associated with the domain name. The tree sub-divides into *zones* beginning at the root zone. A DNS zone may consist of only one domain, or may consist of many domains and sub-domains, depending on the administrative authority delegated to the manager.

Administrative responsibility over any zone may be divided by creating additional zones. Authority is said to be *delegated* for a portion of the old space, usually in the form of sub-domains, to another nameserver and administrative entity. The old zone ceases to be authoritative for the new zone.



The hierarchical Domain Name System, organized into zones, each served by a name server

Domain name syntax

The definitive descriptions of the rules for forming domain names appear in RFC 1035, RFC 1123, and RFC 2181. A domain name consists of one or more parts, technically called *labels*, that are conventionally concatenated, and delimited by dots, such as `example.com`.

- The right-most label conveys the top-level domain; for example, the domain name `www.example.com` belongs to the top-level domain `com`.
- The hierarchy of domains descends from right to left; each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label `example` specifies a subdomain of the `com` domain, and `www` is a sub domain of `example.com`. This tree of subdivisions may have up to 127 levels.
- Each label may contain up to 63 characters. The full domain name may not exceed a total length of 253 characters in its external dotted-label specification.[10] In the internal binary representation of the DNS the maximum length requires 255 octets of storage.[3] In practice, some domain registries may have shorter limits.
- DNS names may technically consist of any character representable in an octet. However, the allowed formulation of domain names in the DNS root zone, and most other sub domains, uses a preferred format and character set. The characters allowed in a label are a subset of the ASCII character set, and includes the characters *a* through *z*, *A* through *Z*, digits *0* through *9*, and the hyphen. This rule is known as the *LDH rule* (letters, digits, hyphen). Domain names are interpreted in case-independent manner. Labels may not start or end with a hyphen.[11]
- A hostname is a domain name that has at least one IP address associated. For example, the domain names `www.example.com` and `example.com` are also hostnames, whereas the `com` domain is not.

Internationalized domain names

The permitted character set of the DNS prevented the representation of names and words of many languages in their native alphabets or scripts. ICANN has approved the Internationalizing Domain Names in Applications (IDNA) system, which maps Unicode strings into the valid DNS character set using Punycode. In 2009 ICANN approved the installation of IDN country code top-level domains. In addition, many registries of the existing top level domain names (TLD)s have adopted IDNA.

Name servers

The Domain Name System is maintained by a distributed database system, which uses the client-server model. The nodes of this database are the name servers. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root nameservers, the servers to query when looking up (*resolving*) a TLD.

Authoritative name server

An *authoritative* name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator.

An authoritative name server can either be a *master* server or a *slave* server. A master server is a server that stores the original (*master*) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records.

Every DNS zone must be assigned a set of authoritative name servers that are installed in NS records in the parent zone.

When domain names are registered with a domain name registrar, their installation at the domain registry of a top level domain requires the assignment of a *primary* name server and at least one *secondary* name server. The requirement of multiple name servers aims to make the domain still functional even if one name server becomes inaccessible or inoperable.[12] The designation of a primary name server is solely determined by the priority given to the domain name registrar. For this purpose, generally only the fully qualified domain name of the name server is required, unless the servers are contained in the registered domain, in which case the corresponding IP address is needed as well.

Primary name servers are often master name servers, while secondary name server may be implemented as slave servers.

An authoritative server indicates its status of supplying definitive answers, deemed *authoritative*, by setting a software flag (a protocol structure bit), called the *Authoritative Answer* (*AA*) bit in its responses.[5] This flag is usually reproduced prominently in the output of DNS administration query tools (such as dig) to indicate *that the responding name server is an authority for the domain name in question.*[5]

Recursive and caching name server

In principle, authoritative name servers are sufficient for the operation of the Internet. However, with only authoritative name servers operating, every DNS query must start with recursive queries at the root zone of the Domain Name System and each user system must implement resolver software capable of recursive operation.

To improve efficiency, reduce DNS traffic across the Internet, and increase performance in end-user applications, the Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question. Typically, such *caching* DNS servers, also called *DNS caches*, also implement the recursive algorithm necessary to resolve a given name starting with the DNS root through to the authoritative name servers of the queried domain. With this function implemented in the name server, user applications gain efficiency in design and operation.

The combination of DNS caching and recursive functions in a name server is not mandatory; the functions can be implemented independently in servers for special purposes.

Internet service providers typically provide recursive and caching name servers for their customers. In addition, many home networking routers implement DNS caches and recursors to improve efficiency in the local network.

DNS resolvers

The client-side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.

A DNS query may be either a non-recursive query or a recursive query:

- A *non-recursive query* is one in which the DNS server provides a record for a domain for which it is authoritative itself, or it provides a partial result without querying other servers.
- A *recursive query* is one for which the DNS server will fully answer the query (or give an error) by querying other name servers as needed. DNS servers are not required to support recursive queries.

The resolver, or another DNS server acting recursively on behalf of the resolver, negotiates use of recursive service using bits in the query headers.

Resolving usually entails iterating through several name servers to find the needed information. However, some resolvers function more simply by communicating only with a single name server. These simple resolvers (called "stub resolvers") rely on a recursive name server to perform the work of finding information for them.

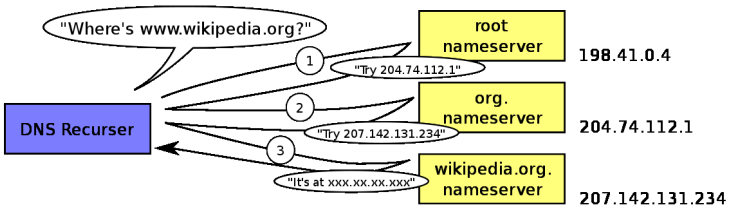
Operation

Address resolution mechanism

Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.

The process entails:

1. A network host is configured with an initial cache (so called *hints*) of the known addresses of the root nameservers. Such a *hint file* is updated periodically by an administrator from a reliable source.



A DNS recursor consults three nameservers to resolve the address www.wikipedia.org.

2. A query to one of the root servers to find the server authoritative for the top-level domain.
3. A query to the obtained TLD server for the address of a DNS server authoritative for the second-level domain.
4. Repetition of the previous step to process each domain name label in sequence, until the final step which returns the IP address of the host sought.

The diagram illustrates this process for the host www.wikipedia.org.

The mechanism in this simple form would place a large operating burden on the root servers, with every search for an address starting by querying one of them. Being as critical as they are to the overall function of the system, such heavy use would create an insurmountable bottleneck for trillions of queries placed every day. In practice caching is used in DNS servers to overcome this problem, and as a result, root nameservers actually are involved with very little of the total traffic.

Circular dependencies and glue records

Name servers in delegations are identified by name, rather than by IP address. This means that a resolving name server must issue another DNS request to find out the IP address of the server to which it has been referred. If the name given in the delegation is a subdomain of the domain for which the delegation is being provided, there is a circular dependency. In this case the nameserver providing the delegation must also provide one or more IP addresses for the authoritative nameserver mentioned in the delegation. This information is called *glue*. The delegating name server provides this glue in the form of records in the *additional section* of the DNS response, and provides the delegation in the *answer section* of the response.

For example, if the authoritative name server for `example.org` is `ns1.example.org`, a computer trying to resolve `www.example.org` first resolves `ns1.example.org`. Since `ns1` is contained in `example.org`, this requires resolving `example.org` first, which presents a circular dependency. To break the dependency, the nameserver for the `org` top level domain includes glue along with the delegation for `example.org`. The glue records are address records that provide IP addresses for `ns1.example.org`. The resolver uses one or more of these IP addresses to query one of domain's authoritative servers, which allows it to complete the DNS query.

Record caching

Because of the large volume of DNS requests generated for the public Internet, the designers wished to provide a mechanism to reduce the load on individual DNS servers. To this end, the DNS resolution process allows for *caching* of records for a period of time after an answer. This entails the local recording and subsequent consultation of the copy instead of initiating a new request upstream. The time for which a resolver caches a DNS response is determined by a value called the time to live (TTL) associated with every record. The TTL is set by the administrator of the DNS server handing out the authoritative response. The period of validity may vary from just seconds to days or even weeks.

As a noteworthy consequence of this distributed and caching architecture, changes to DNS records do not propagate throughout the network immediately, but require all caches to expire and refresh after the TTL. RFC 1912 conveys basic rules for determining appropriate TTL values.

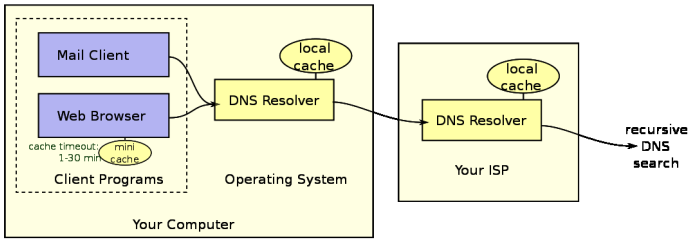
Some resolvers may override TTL values, as the protocol supports caching for up to 68 years or no caching at all. Negative caching, i.e. the caching of the fact of non-existence of a record, is determined by name servers authoritative for a zone which must include the Start of Authority (SOA) record when reporting no data of the requested type exists. The value of the *MINIMUM* field of the SOA record and the TTL of the SOA itself is used to establish the TTL for the negative answer.

Reverse lookup

A reverse lookup is a query of the DNS for domain names when the IP address is known. Multiple domain names may be associated with an IP address. The DNS stores IP addresses in the form of domain names as specially formatted names in pointer (`PTR`) records within the infrastructure top-level domain arpa. For IPv4, the domain is `in-addr.arpa`. For IPv6, the reverse lookup domain is `ip6.arpa`. The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.

When performing a reverse lookup, the DNS client converts the address into these formats, and then queries the name for a PTR record following the delegation chain as for any DNS query. For example, the IPv4 address `208.80.152.2` is represented as a DNS name as `2.152.80.208.in-addr.arpa`. The DNS resolver begins by querying the root servers, which point to ARIN's servers for the `208.in-addr.arpa` zone. From there the Wikimedia servers are assigned for `152.80.208.in-addr.arpa`, and the PTR lookup completes by querying the wikimedia nameserver for `2.152.80.208.in-addr.arpa`, which results in an authoritative response.

Client lookup



DNS resolution sequence

Users generally do not communicate directly with a DNS resolver. Instead DNS resolution takes place transparently in applications programs such as web browsers, e-mail clients, and other Internet applications. When an application makes a request that requires a domain name lookup, such programs send a resolution request to the DNS resolver in the local operating system, which in turn handles the communications required.

The DNS resolver will almost invariably have a cache (see above) containing recent lookups. If the cache can provide the answer to the request, the resolver will return the value in the cache to the program that made the request. If the cache does not contain the answer, the resolver will send the request to one or more designated DNS servers. In the case of most home users, the Internet service provider to which the machine connects will usually supply this DNS server: such a user will either have configured that server's address manually or allowed DHCP to set it; however, where systems administrators have configured systems to use their own DNS servers, their DNS resolvers point to separately maintained nameservers of the organization. In any event, the name server thus queried will follow the process outlined above, until it either successfully finds a result or does not. It then returns its results to the DNS resolver; assuming it has found a result, the resolver duly caches that result for future use, and hands the result back to the software which initiated the request.

Broken resolvers

An additional level of complexity emerges when resolvers violate the rules of the DNS protocol. A number of large ISPs have configured their DNS servers to violate rules (presumably to allow them to run on less-expensive hardware than a fully compliant resolver), such as by disobeying TTLs, or by indicating that a domain name does not exist just because one of its name servers does not respond.[13]

As a final level of complexity, some applications (such as web-browsers) also have their own DNS cache, in order to reduce the use of the DNS resolver library itself. This practice can add extra difficulty when debugging DNS issues, as it obscures the freshness of data, and/or what data comes from which cache. These caches typically use very short caching times—on the order of one minute.

Internet Explorer represents a notable exception: versions up to IE 3.x cache DNS records for 24 hours by default. Internet Explorer 4.x and later versions (up to IE 8) decrease the default time out value to half an hour, which may be changed in corresponding registry keys.[14]

Other applications

The system outlined above provides a somewhat simplified scenario. The Domain Name System includes several other functions:

- Hostnames and IP addresses do not necessarily match on a one-to-one basis. Multiple hostnames may correspond to a single IP address: combined with virtual hosting, this allows a single machine to serve many web sites. Alternatively a single hostname may correspond to many IP addresses: this can facilitate fault tolerance and load distribution, and also allows a site to move physical location seamlessly.
- There are many uses of DNS besides translating names to IP addresses. For instance, Mail transfer agents use DNS to find out where to deliver e-mail for a particular address. The domain to mail exchanger mapping provided by MX records accommodates another layer of fault tolerance and load distribution on top of the name to IP address mapping.
- E-mail Blacklists: The DNS system is used for efficient storage and distribution of IP addresses of blacklisted e-mail hosts. The usual method is putting the IP address of the subject host into the sub-domain of a higher level domain name, and resolve that name to different records to indicate a positive or a negative. Here is a hypothetical example blacklist:
  - 102.3.4.5 is blacklisted => Creates 5.4.3.102.blacklist.example and resolves to 127.0.0.1
  - 102.3.4.6 is not => 6.4.3.102.blacklist.example is not found, or default to 127.0.0.2
  - E-mail servers can then query blacklist.example through the DNS mechanism to find out if a specific host connecting to them is in the blacklist. Today many of such blacklists, either free or subscription-based, are available mainly for use by email administrators and anti-spam software.
- Software Updates: many anti-virus and commercial software now use the DNS system to store version numbers of the latest software updates so client computers do not need to connect to the update servers every time. For these types of applications, the cache time of the DNS records are usually shorter.
- Sender Policy Framework and DomainKeys, instead of creating their own record types, were designed to take advantage of another DNS record type, the TXT record.
- To provide resilience in the event of computer failure, multiple DNS servers are usually provided for coverage of each domain, and at the top level, thirteen very powerful root servers exist, with additional "copies" of several of them distributed worldwide via Anycast.
- Dynamic DNS (sometimes called DDNS) allows clients to update their DNS entry as their IP address changes, as it does, for example, when moving between ISPs or mobile hot spots.

Protocol details

DNS primarily uses User Datagram Protocol (UDP) on port number 53 to serve requests.[5] DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server. The Transmission Control Protocol (TCP) is used when the response data size exceeds 512 bytes, or for tasks such as zone transfers. Some resolver implementations use TCP for all queries.

DNS resource records

Further information: List of DNS record types

A Resource Record (RR) is the basic data element in the domain name system. Each record has a type (A, MX, etc.), an expiration time limit, a class, and some type-specific data. Resource records of the same type define a resource record set (RRset). The order of resource records in a set, returned by a resolver to an application, is undefined, but often servers implement round-robin ordering to achieve load balancing. DNSSEC, however, works on complete resource record sets in a canonical order.

When sent over an IP network, all records use the common format specified in RFC 1035:[15]

RR (Resource record) fields

Field	Description	Length (octets)
NAME	Name of the node to which this record pertains	(variable)
TYPE	Type of RR in numeric form (e.g. 15 for MX RRs)	2
CLASS	Class code	2
TTL	Count of seconds that the RR stays valid (The maximum is 2^{31}-1, which is about 68 years.)	4
RDLENGTH	Length of RDATA field	2
RDATA	Additional RR-specific data	(variable)

NAME is the fully qualified domain name of the node in the tree. On the wire, the name may be shortened using label compression where ends of domain names mentioned earlier in the packet can be substituted for the end of the current domain name.

TYPE is the record type. It indicates the format of the data and it gives a hint of its intended use. For example, the *A* record is used to translate from a domain name to an IPv4 address, the *NS* record lists which name servers can answer lookups on a DNS zone, and the *MX* record specifies the mail server used to handle mail for a domain specified in an e-mail address (see also List of DNS record types).

RDATA is data of type-specific relevance, such as the IP address for address records, or the priority and hostname for MX records. Well known record types may use label compression in the RDATA field, but "unknown" record types must not (RFC 3597).

The *CLASS* of a record is set to `IN` (for *Internet*) for common DNS records involving Internet hostnames, servers, or IP addresses. In addition, the classes Chaos (`CH`) and Hesiod (`HS`) exist.[16] Each class is an independent name space with potentially different delegations of DNS zones.

In addition to resource records defined in a zone file, the domain name system also defines several request types that are used only in communication with other DNS nodes (*on the wire*), such as when performing zone transfers (AXFR/IXFR) or for EDNS (OPT).

Wildcard DNS records

The domain name system supports *wildcard domain names* which are names that start with the *asterisk label*, '*', e.g., `*.example`.[3] [17] DNS records belonging to wildcard domain names specify rules for generating resource records within a single DNS zone by substituting whole labels with matching components of the query name, including any specified descendants. For example, in the DNS zone *x.example*, the following configuration specifies that all subdomains (including subdomains of subdomains) of *x.example* use the mail exchanger *a.x.example*. The records for *a.x.example* are needed to specify the mail exchanger. As this has the result of excluding this domain name and its subdomains from the wildcard matches, all subdomains of *a.x.example* must be defined in a separate wildcard statement.

The role of wildcard records was refined in RFC 4592, because the original definition in RFC 1034 was incomplete and resulted in misinterpretations by implementers.[17]

Protocol extensions

The original DNS protocol had limited provisions for extension with new features. In 1999, Paul Vixie published in RFC 2671 an extension mechanism, called Extension mechanisms for DNS (EDNS) that introduced optional protocol elements without increasing overhead when not in use. This was accomplished through the `OPT` pseudo-resource record that only exists in wire transmissions of the protocol, but not in any zone files. Initial extensions were also suggested (EDNS0), such as increasing the DNS message size in UDP datagrams.

Dynamic zone updates

Dynamic DNS updates use the `UPDATE` DNS opcode to add or remove resource records dynamically from a zone data base maintained on an authoritative DNS server. The feature is described in RFC 2136. This facility is useful to register network clients into the DNS when they boot or become otherwise available on the network. Since a booting client may be assigned a different IP address each time from a DHCP server, it is not possible to provide static DNS assignments for such clients.

Security issues

Originally, security concerns were not major design considerations for DNS software or any software for deployment on the early Internet, as the network was not open for participation by the general public. However, the expansion of the Internet into the commercial sector in the 1990s changed the requirements for security measures to protect data integrity and user authentication.

Several vulnerability issues were discovered and exploited by malicious users. One such issue is DNS cache poisoning, in which data is distributed to caching resolvers under the pretense of being an authoritative origin server, thereby polluting the data store with potentially false information and long expiration times (time-to-live). Subsequently, legitimate application requests may be redirected to network hosts operated with malicious intent.

DNS responses are traditionally not cryptographically signed, leading to many attack possibilities; the Domain Name System Security Extensions (DNSSEC) modify DNS to add support for cryptographically signed responses. Several extensions have been devised to secure zone transfers as well.

Some domain names may be used to achieve spoofing effects. For example, paypal.com and paypa1.com are different names, yet users may be unable to distinguish them in a graphical user interface depending on the user's chosen typeface. In many fonts the letter *l* and the numeral *1* look very similar or even identical. This problem is acute in systems that support internationalized domain names, since many character codes in ISO 10646, may appear identical on typical computer screens. This vulnerability is occasionally exploited in phishing.[18]

Techniques such as forward-confirmed reverse DNS can also be used to help validate DNS results.

Domain name registration

The right to use a domain name is delegated by domain name registrars which are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN), the organization charged with overseeing the name and number systems of the Internet. In addition to ICANN, each top-level domain (TLD) is maintained and serviced technically by an administrative organization, operating a registry. A registry is responsible for maintaining the database of names registered within the TLD it administers. The registry receives registration information from each domain name registrar authorized to assign names in the corresponding TLD and publishes the information using a special service, the whois protocol.

ICANN publishes the complete list of TLD registries and domain name registrars. Registrant information associated with domain names is maintained in an online database accessible with the WHOIS service. For most of the more than 240 country code top-level domains (ccTLDs), the domain registries maintain the WHOIS (Registrant, name servers, expiration dates, etc.) information. For instance, DENIC, Germany NIC, holds the DE domain data. Since about 2001, most gTLD registries have adopted this so-called *thick* registry approach, i.e. keeping the WHOIS data in central registries instead of registrar databases.

For `COM` and `NET` domain names, a *thin* registry model is used: the domain registry (e.g. VeriSign) holds basic WHOIS (registrar and name servers, etc.) data. One can find the detailed WHOIS (registrant, name servers, expiry dates, etc.) at the registrars.

Some domain name registries, often called *network information centers* (NIC), also function as registrars to end-users. The major generic top-level domain registries, such as for the `COM`, `NET`, `ORG`, `INFO` domains, use a registry-registrar model consisting of many domain name registrars[19] [20] In this method of management, the registry only manages the domain name database and the relationship with the registrars. The *registrants* (users of a domain name) are customers of the registrar, in some cases through additional layers of resellers.

Internet standards

The Domain Name System is defined by Request for Comments (RFC) documents published by the Internet Engineering Task Force (Internet standards). The following is a list of RFCs that define the DNS protocol.

- RFC 920, *Domain Requirements* – Specified original top-level domains
- RFC 1032, *Domain Administrators Guide*
- RFC 1033, *Domain Administrators Operations Guide*
- RFC 1034, *Domain Names - Concepts and Facilities*
- RFC 1035, *Domain Names - Implementation and Specification*
- RFC 1101, *DNS Encodings of Network Names and Other Types*
- RFC 1123, *Requirements for Internet Hosts—Application and Support*
- RFC 1178, *Choosing a Name for Your Computer* (FYI 5)
- RFC 1183, *New DNS RR Definitions*
- RFC 1591, *Domain Name System Structure and Delegation* (Informational)
- RFC 1912, *Common DNS Operational and Configuration Errors*
- RFC 1995, *Incremental Zone Transfer in DNS*
- RFC 1996, *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*
- RFC 2100, *The Naming of Hosts* (Informational)
- RFC 2136, *Dynamic Updates in the domain name system (DNS UPDATE)*
- RFC 2181, *Clarifications to the DNS Specification*
- RFC 2182, *Selection and Operation of Secondary DNS Servers*
- RFC 2308, *Negative Caching of DNS Queries (DNS NCACHE)*
- RFC 2317, *Classless IN-ADDR.ARPA delegation* (BCP 20)

- RFC 2671, *Extension Mechanisms for DNS (EDNS0)*
- RFC 2672, *Non-Terminal DNS Name Redirection*
- RFC 2845, *Secret Key Transaction Authentication for DNS (TSIG)*
- RFC 3225, *Indicating Resolver Support of DNSSEC*
- RFC 3226, *DNSSEC and IPv6 A6 aware server/resolver message size requirements*
- RFC 3597, *Handling of Unknown DNS Resource Record (RR) Types*
- RFC 3696, *Application Techniques for Checking and Transformation of Names* (Informational)
- RFC 4343, *Domain Name System (DNS) Case Insensitivity Clarification*
- RFC 4592, *The Role of Wildcards in the Domain Name System*
- RFC 4635, *HMAC SHA TSIG Algorithm Identifiers*
- RFC 4892, *Requirements for a Mechanism Identifying a Name Server Instance* (Informational)
- RFC 5001, *DNS Name Server Identifier (NSID) Option*
- RFC 5452, *Measures for Making DNS More Resilient against Forged Answers*
- RFC 5625, *DNS Proxy Implementation Guidelines* (BCP 152)
- RFC 5890, *Internationalized Domain Names for Applications (IDNA):Definitions and Document Framework*
- RFC 5891, *Internationalized Domain Names in Applications (IDNA): Protocol*
- RFC 5892, *The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)*
- RFC 5893, *Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)*
- RFC 5894, *Internationalized Domain Names for Applications (IDNA):Background, Explanation, and Rationale* (Informational)
- RFC 5895, *Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008* (Informational)
- RFC 6195, *Domain Name System (DNS) IANA Considerations* (BCP 42)

Security

- RFC 4033, *DNS Security Introduction and Requirements*
- RFC 4034, *Resource Records for the DNS Security Extensions*
- RFC 4035, *Protocol Modifications for the DNS Security Extensions*
- RFC 4509, *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records*
- RFC 4470, *Minimally Covering NSEC Records and DNSSEC On-line Signing*
- RFC 5011, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*
- RFC 5155, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*
- RFC 5702, *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC*
- RFC 5910, *Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)*
- RFC 5933, *Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC*

References

[1] Mockapetris, Paul (2004-01-02). "Letting DNS Loose" (http://www.circleid.com/posts/letting_dns_loose/). CircleID. . "RFID tags, UPC codes, International characters in email addresses and host names, and a variety of other identifiers could all go into DNS [...] — it's ready to carry arbitrary identifiers."

[2] Mockapetris, Paul (April 1989). "RFC 1101: DNS Encoding of Network Names and Other Types". p. 1. "The DNS is extensible and can be used for a virtually unlimited number of data types, name spaces, etc."

[3] RFC 1034, *Domain Names - Concepts and Facilities*, P. Mockapetris, The Internet Society (November 1987)

[4] RFC 781, *Internet Protocol - DARPA Internet Program Protocol Specification*, Information Sciences Institute, J. Postel (Ed.), The Internet Society (September 1981)

[5] RFC 1035, *Domain Names - Implementation and Specification*, P. Mockapetris, The Internet Society (November 1987)

[6] RFC 3467, *Role of the Domain Name System (DNS)*, J.C. Klensin, J. Klensin (February 2003)

[7] Cricket Liu, Paul Albitz (2006). *DNS and BIND* (http://oreilly.com/catalog/9780596100575) (5th ed.). O'Reilly. p. 3. .

[8] Douglas Brian Terry, Mark Painter, David W. Riggle and Songnian Zhou, *The Berkeley Internet Name Domain Server* (http://www.eecs.berkeley.edu/Pubs/TechRpts/1984/5957.html), Proceedings USENIX Summer Conference, Salt Lake City, Utah, June 1984, pages 23–31.

[9] "DNS Server Survey" (http://mydns.bboy.net/survey/). .

[10] RFC 2181, *Clarifications to the DNS Specification*, R. Elz, R. Bush (July 1997)

[11] RFC 3696, *Application Techniques for Checking and Transformation of Names*, J.C. Klensin, J. Klensin

[12] "Name Server definition at techterms.com" (http://www.techterms.com/definition/nameserver). .

[13] "Providers ignoring DNS TTL ?" (http://ask.slashdot.org/article.pl?sid=05/04/18/198259). Slashdot. 2005. . Retrieved 2009-01-03.

[14] "How Internet Explorer uses the cache for DNS host entries" (http://support.microsoft.com/default.aspx?scid=KB;en-us;263558). Microsoft Corporation. 2004. . Retrieved 2010-07-25.

[15] RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), Section 3

[16] RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), p. 11

[17] RFC 4592, *The Role of Wildcards in the Domain Name System*, E. Lewis (July 2006)

[18] APWG. "Global Phishing Survey: Domain Name Use and Trends in 1H2010." 10/15/2010 apwg.org (http://www.apwg.org/reports/APWG_GlobalPhishingSurvey_1H2010.pdf.)

[19] ICANN accredited registrars (http://www.icann.org/registrars/accredited-list.html)

[20] VeriSign COM and NET registry (http://www.verisign.com/information-services/naming-services/com-net-registry/page_002166.html)

External links

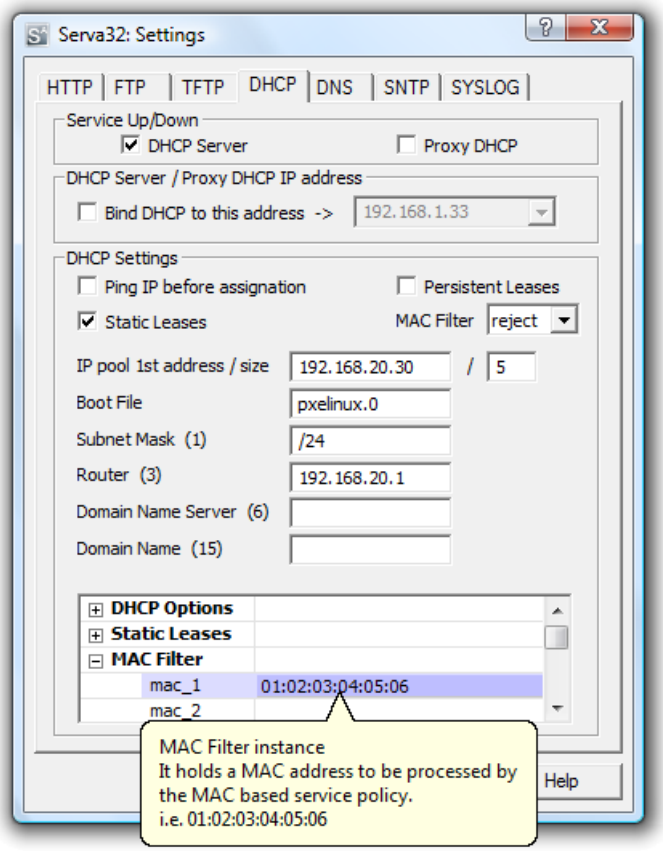
- Vixie, Paul (2007-05-04). "DNS Complexity – Although it contains just a few simple rules, DNS has grown into an enormously complex system." (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=481). ACM Queue.
- Zytrax.com (http://www.zytrax.com/books/dns/), Open Source Guide – DNS for Rocket Scientists, an on-line technical.
- Domain Name System (http://www.microsoft.com/dns) on Microsoft TechNet

Dynamic Host Configuration Protocol

The **Dynamic Host Configuration Protocol** (**DHCP**) is a network configuration protocol for hosts on Internet Protocol (IP) networks. Computers that are connected to IP networks must be configured before they can communicate with other hosts. The most essential information needed is an IP address, and a default route and routing prefix. DHCP eliminates the manual task by a network administrator. It also provides a central database of devices that are connected to the network and eliminates duplicate resource assignments.

In addition to IP addresses, DHCP also provides other configuration information, particularly the IP addresses of local caching DNS resolvers, network boot servers, or other service hosts.

DHCP is used for IPv4 as well as IPv6. While both versions perform much the same purpose, the details of the protocol for IPv4 and IPv6 are sufficiently different that they may be considered separate protocols.[1]



A DHCP Server settings tab

Hosts that do not use DHCP for address configuration may still use it to obtain other configuration information. Alternatively, IPv6 hosts may use stateless address autoconfiguration. IPv4 hosts may use link-local addressing to achieve limited local connectivity.

History

DHCP was first defined as a standards track protocol in RFC 1531 in October 1993, as an extension to the Bootstrap Protocol (BOOTP). The motivation for extending BOOTP was that BOOTP required manual intervention to add configuration information for each client, and did not provide a mechanism for reclaiming disused IP addresses.

Many worked to clarify the protocol as it gained popularity, and in 1997 RFC 2131 was released, and remains as of 2011 the standard for IPv4 networks. DHCPv6 is documented in RFC 3315. RFC 3633 added a DHCPv6 mechanism for prefix delegation. DHCPv6 was further extended to provide configuration information to clients configured using stateless address autoconfiguration in RFC 3736.

The BOOTP protocol itself was first defined in RFC 951 as a replacement for the Reverse Address Resolution Protocol RARP. The primary motivation for replacing RARP with BOOTP was that RARP was a data link layer protocol. This made implementation difficult on many server platforms, and required that a server be present on each individual network link. BOOTP introduced the innovation of a *relay agent*, which allowed the forwarding of BOOTP packets off the local network using standard IP routing, thus one central BOOTP server could serve hosts on many IP subnets.[2]

Technical overview

Dynamic Host Configuration Protocol automates network-parameter assignment to network devices from one or more DHCP servers. Even in small networks, DHCP is useful because it makes it easy to add new machines to the network.

When a DHCP-configured client (a computer or any other network-aware device) connects to a network, the DHCP client sends a broadcast query requesting necessary information from a DHCP server. The DHCP server manages a pool of IP addresses and information about client configuration parameters such as default gateway, domain name, the name servers, other servers such as time servers, and so forth. On receiving a valid request, the server assigns the computer an IP address, a lease (length of time the allocation is valid), and other IP configuration parameters, such as the subnet mask and the default gateway. The query is typically initiated immediately after booting, and must complete before the client can initiate IP-based communication with other hosts.

Depending on implementation, the DHCP server may have three methods of allocating IP-addresses:

- *dynamic allocation*: A network administrator assigns a range of IP addresses to DHCP, and each client computer on the LAN is configured to request an IP address from the DHCP server during network initialization. The request-and-grant process uses a lease concept with a controllable time period, allowing the DHCP server to reclaim (and then reallocate) IP addresses that are not renewed.
- *automatic allocation*: The DHCP server permanently assigns a free IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had.
- *static allocation*: The DHCP server allocates an IP address based on a table with MAC address/IP address pairs, which are manually filled in (perhaps by a network administrator). Only requesting clients with a MAC address listed in this table will be allocated an IP address. This feature (which is not supported by all DHCP servers) is variously called *Static DHCP Assignment* (by DD-WRT), *fixed-address* (by the dhcpd documentation), *Address Reservation* (by Netgear), *DHCP reservation* or *Static DHCP* (by Cisco/Linksys), and *IP reservation* or *MAC/IP binding* (by various other router manufacturers).

Technical details

DHCP uses the same two ports assigned by IANA for BOOTP: destination UDP port 67 for sending data to the server, and UDP port 68 for data to the client. DHCP communications are connectionless in nature.

DHCP operations fall into four basic phases: IP discovery, IP lease offer, IP request, and IP lease acknowledgement.

DHCP clients and servers on the same subnet communicate via UDP broadcasts, initially. If the client and server are on different subnets, a DHCP Helper or DHCP Relay Agent may be used. Clients requesting renewal of an existing lease may communicate directly via UDP unicast, since the client already has an established IP address at that point.

DHCP discovery

The client broadcasts messages on the physical subnet to discover available DHCP servers. Network administrators can configure a local router to forward DHCP packets to a DHCP server from a different subnet. This client-implementation creates a User Datagram Protocol (UDP) packet with the broadcast destination of 255.255.255.255 or the specific subnet broadcast address.

A DHCP client can also request its last-known IP address (in the example below, 192.168.1.100). If the client remains connected to a network for which this IP is valid, the server may grant the request. Otherwise, it depends whether the server is set up as authoritative or not. An authoritative server will deny the request, making the client ask for a new IP address immediately. A non-authoritative server simply ignores the request, leading to an implementation-dependent timeout for the client to give up on the request and ask for a new IP address.

DHCPDISCOVER

UDP Src=0.0.0.0 sPort=68 Dest=255.255.255.255 dPort=67			
OP	**HTYPE**	**HLEN**	**HOPS**
0x01	0x01	0x06	0x00
XID			
0x3903F326			
SECS		**FLAGS**	
0x0000		0x0000	
CIADDR (Client IP Address)			
0x00000000			
YIADDR (Your IP Address)			
0x00000000			
SIADDR (Server IP Address)			
0x00000000			
GIADDR (Gateway IP Address)			
0x00000000			
CHADDR (Client Hardware Address)			
0x00053C04			
0x8D590000			
0x00000000			
0x00000000			
192 octets of 0s. BOOTP legacy			
Magic Cookie			
0x63825363			
DHCP Options			
DHCP option 53: DHCP Discover			
DHCP option 50: 192.168.1.100 requested			
DHCP option 55: Parameter Request List: Request Subnet Mask (1), Router (3), Domain Name (15), Domain Name Server (6)			

DHCP offer

When a DHCP server receives an IP lease request from a client, it reserves an IP address for the client and extends an IP lease offer by sending a DHCPOFFER message to the client. This message contains the client's MAC address, the IP address that the server is offering, the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

The server determines the configuration based on the client's hardware address as specified in the CHADDR (Client Hardware Address) field. Here the server, 192.168.1.1, specifies the IP address in the YIADDR (Your IP Address) field.

DHCPOFFER

UDP Src=192.168.1.1 sPort=67 Dest=255.255.255.255 dPort=68			
OP	**HTYPE**	**HLEN**	**HOPS**
0x02	0x01	0x06	0x00
XID			
0x3903F326			
SECS		**FLAGS**	
0x0000		0x0000	
CIADDR (Client IP Address)			
0x00000000			
YIADDR (Your IP Address)			
0xC0A80164			
SIADDR (Server IP Address)			
0xC0A80101			
GIADDR (Gateway IP Address)			
0x00000000			
CHADDR (Client Hardware Address)			
0x00053C04			
0x8D590000			
0x00000000			
0x00000000			
192 octets of 0s. BOOTP legacy			
Magic Cookie			
0x63825363			
DHCP Options			
DHCP option 53: DHCP Offer			
DHCP option 1: 255.255.255.0 subnet mask			
DHCP option 3: 192.168.1.1 router			
DHCP option 51: 86400s (1 day) IP lease time			
DHCP option 54: 192.168.1.1 DHCP server			
DHCP option 6: DNS servers 9.7.10.15, 9.7.10.16, 9.7.10.18			

DHCP request

A client can receive DHCP offers from multiple servers, but it will accept only one DHCP offer and broadcast a DHCP request message. Based on the Transaction ID field in the request, servers are informed whose offer the client has accepted. When other DHCP servers receive this message, they withdraw any offers that they might have made to the client and return the offered address to the pool of available addresses. The DHCP request message is broadcast, instead of being unicast to a particular DHCP server, because the DHCP client has still not received an IP address. Also, this way one message can let all other DHCP servers know that another server will be supplying the

IP address without missing any of the servers with a series of unicast messages.

DHCPREQUEST

UDP Src=0.0.0.0 sPort=68 Dest=255.255.255.255 dPort=67			
OP	**HTYPE**	**HLEN**	**HOPS**
0x01	0x01	0x06	0x00
XID			
0x3903F326			
SECS		**FLAGS**	
0x0000		0x0000	
CIADDR (Client IP Address)			
0x00000000			
YIADDR (Your IP Address)			
0x00000000			
SIADDR (Server IP Address)			
0xC0A80101			
GIADDR (Gateway IP Address)			
0x00000000			
CHADDR (Client Hardware Address)			
0x00053C04			
0x8D590000			
0x00000000			
0x00000000			
192 octets of 0s. BOOTP legacy			
Magic Cookie			
0x63825363			
DHCP Options			
DHCP option 53: DHCP Request			
DHCP option 50: 192.168.1.100 requested			
DHCP option 54: 192.168.1.1 DHCP server.			

DHCP acknowledgement

When the DHCP server receives the DHCPREQUEST message from the client, the configuration process enters its final phase. The acknowledgement phase involves sending a DHCPACK packet to the client. This packet includes the lease duration and any other configuration information that the client might have requested. At this point, the IP configuration process is completed.

The protocol expects the DHCP client to configure its network interface with the negotiated parameters.

DHCPACK

UDP Src=192.168.1.1 sPort=67 Dest=255.255.255.255 dPort=68				
OP	**HTYPE**	**HLEN**	**HOPS**	
0x02	0x01	0x06	0x00	
XID				
0x3903F326				
SECS			**FLAGS**	
0x0000		0x0000		
CIADDR (Client IP Address)				
0x00000000				
YIADDR (Your IP Address)				
0xC0A80164				
SIADDR (Server IP Address)				
0xC0A80101				
GIADDR (Gateway IP Address switched by relay)				
0x00000000				
CHADDR (Client Hardware Address)				
0x00053C04				
0x8D590000				
0x00000000				
0x00000000				
192 octets of 0s. BOOTP legacy				
Magic Cookie				
0x63825363				
DHCP Options				
DHCP option 53: DHCP ACK				
DHCP option 1: 255.255.255.0 subnet mask				
DHCP option 3: 192.168.1.1 router				
DHCP option 51: 86400s (1 day) IP lease time				
DHCP option 54: 192.168.1.1 DHCP server				
DHCP option 6: DNS servers 9.7.10.15, 9.7.10.16, 9.7.10.18				

After the client obtains an IP address, the client may use the Address Resolution Protocol (ARP) to prevent IP conflicts caused by overlapping address pools of DHCP servers.

DHCP information

A DHCP client may request more information than the server sent with the original DHCPOFFER. The client may also request repeat data for a particular application. For example, browsers use *DHCP Inform* to obtain web proxy settings via WPAD. Such queries do not cause the DHCP server to refresh the IP expiry time in its database.

DHCP releasing

The client sends a request to the DHCP server to release the DHCP information and the client deactivates its IP address. As client devices usually do not know when users may unplug them from the network, the protocol does not mandate the sending of *DHCP Release*.

Client configuration parameters in DHCP

A DHCP server can provide optional configuration parameters to the client. RFC 2132 describes the available DHCP options defined by Internet Assigned Numbers Authority (IANA) - DHCP and BOOTP PARAMETERS [3].

A DHCP client can select, manipulate and overwrite parameters provided by a DHCP server.[4]

Options

An option exists to identify the vendor and functionality of a DHCP client. The information is a variable-length string of characters or octets which has a meaning specified by the vendor of the DHCP client. One method that a DHCP client can utilize to communicate to the server that it is using a certain type of hardware or firmware is to set a value in its DHCP requests called the Vendor Class Identifier (VCI) (Option 60). This method allows a DHCP server to differentiate between the two kinds of client machines and process the requests from the two types of modems appropriately. Some types of set-top boxes also set the VCI (Option 60) to inform the DHCP server about the hardware type and functionality of the device. The value that this option is set to give the DHCP server a hint about any required extra information that this client needs in a DHCP response.

DHCP Relaying

In small networks, where only one IP subnet is being managed, DHCP clients communicate directly with DHCP servers. However, DHCP servers can also provide IP addresses for multiple subnets. In this case, a DHCP client that has not yet acquired an IP address cannot communicate directly with the DHCP server using IP routing, because it doesn't have a routable IP address, nor does it know the IP address of a router. In order to allow DHCP clients on subnets not directly served by DHCP servers to communicate with DHCP servers, DHCP relay agents can be installed on these subnets. The DHCP client broadcasts on the local link; the relay agent receives the broadcast and transmits it to one or more DHCP servers using unicast. The relay agent stores its own IP address in the GIADDR field of the DHCP packet. The DHCP server uses the GIADDR to determine the subnet on which the relay agent received the broadcast, and allocates an IP address on that subnet. When the DHCP server replies to the client, it sends the reply to the GIADDR address, again using unicast. The relay agent then retransmits the response on the local network.

Reliability

The DHCP protocol provides reliability in several ways: periodic renewal, rebinding, and failover. DHCP clients are allocated leases that last for some period of time. Clients begin to attempt to renew their leases once half the lease interval has expired. They do this by sending a unicast DHCPREQUEST message to the DHCP server that granted the original lease. If that server is down or unreachable, it will fail to respond to the DHCPREQUEST. However, the DHCPREQUEST will be repeated by the client from time to time, so when the DHCP server comes back up or becomes reachable again, the DHCP client will succeed in contacting it, and renew its lease.

If the DHCP server is unreachable for an extended period of time, the DHCP client will attempt to rebind, by broadcasting its DHCPREQUEST rather than unicasting it. Because it is broadcast, the DHCPREQUEST message will reach all available DHCP servers. If some other DHCP server is able to renew the lease, it will do so at this time.

In order for rebinding to work, when the client successfully contacts a backup DHCP server, that server must have accurate information about the client's binding. Maintaining accurate binding information between two servers is a complicated problem; if both servers are able to update the same lease database, there must be a mechanism to avoid conflicts between updates on the independent servers. A standard for implementing fault-tolerant DHCP servers was developed at the Internet Engineering Task Force.[5] [6]

If rebinding fails, the lease will eventually expire. When the lease expires, the client must stop using the IP address granted to it in its lease. At that time, it will restart the DHCP process from the beginning by broadcasting a DHCPDISCOVER message. Since its lease has expired, it will accept any IP address offered to it. Once it has a new IP address, presumably from a different DHCP server, it will once again be able to use the network. However, since its IP address has changed, any ongoing connections will be broken.

Security

The base DHCP protocol does not include any mechanism for authentication.[7] Because of this, it is vulnerable to a variety of attacks. These attacks fall into three main categories:

- Unauthorized DHCP servers providing false information to clients.[8]
- Unauthorized clients gaining access to resources.[8]
- Resource exhaustion attacks from malicious DHCP clients.[8]

Because the client has no way to validate the identity of a DHCP server, unauthorized DHCP servers can be operated on networks, providing incorrect information to DHCP clients. This can serve either as a denial-of-service attack, preventing the client from gaining access to network connectivity, or as a man-in-the-middle attack. Because the DHCP server provides the DHCP client with server IP addresses, such as the IP address of one or more DNS servers,[8] an attacker can convince a DHCP client to do its DNS lookups through its own DNS server, and can therefore provide its own answers to DNS queries from the client.[9] This in turn allows the attacker to redirect network traffic through itself, allowing it to eavesdrop on connections between the client and network servers it contacts, or to simply replace those network servers with its own.[9]

Because the DHCP server has no secure mechanism for authenticating the client, clients can gain unauthorized access to IP addresses by presenting credentials, such as client identifiers, that belong to other DHCP clients. This also allows DHCP clients to exhaust the DHCP server's store of IP addresses—by presenting new credentials each time it asks for an address, the client can consume all the available IP addresses on a particular network link, preventing other DHCP clients from getting service.

DHCP does provide some mechanisms for mitigating these problems. The Relay Agent Information Option protocol extension (RFC 3046) allows network operators to attach tags to DHCP messages as these messages arrive on the network operator's trusted network. This tag is then used as an authorization token to control the client's access to network resources. Because the client has no access to the network upstream of the relay agent, the lack of

authentication does not prevent the DHCP server operator from relying on the authorization token.[7]

Another extension, Authentication for DHCP Messages (RFC 3118), provides a mechanism for authenticating DHCP messages. Unfortunately RFC 3118 has not seen widespread adoption because of the problems of managing keys for large numbers of DHCP clients.[10]

Notes

[1] Ralph Droms; Ted Lemon (2003). *The DHCP Handbook*. SAMS Publishing. p. 436. ISBN 0-672-32327-3.

[2] Bill Croft; John Gilmore (September 1985). "RFC 951 - Bootstrap Protocol" (http://tools.ietf.org/html/rfc951#section-6). *Network Working Group*. .

[3] http://www.iana.org/assignments/bootp-dhcp-parameters

[4] In Unix-like systems this client-level refinement typically takes place according to the values in a `/etc/dhclient.conf` configuration file.

[5] Droms, Ralph; Kinnear, Kim; Stapp, Mark; Volz, Bernie; Gonczi, Steve; Rabil, Greg; Dooley, Michael; Kapur, Arun (March 2003). *DHCP Failover Protocol* (http://tools.ietf.org/html/draft-ietf-dhc-failover-12). IETF. I-D draft-ietf-dhc-failover-12. . Retrieved May 09, 2010.

[6] The IETF proposal provided a mechanism whereby two servers could remain loosely in sync with each other in such a way that even in the event of a total failure of one server, the other server could recover the lease database and continue operating. Due to the length and complexity of the specification, it was never published as a standard; however, the techniques described in the specification are in wide use, with one open source implementation in the ISC DHCP server as well as several commercial implementations.

[7] Michael Patrick (January 2001). "RFC 3046 - DHCP Relay Agent Information Option" (http://tools.ietf.org/html/rfc3046#section-7). *Network Working Group*. .

[8] Ralph Droms (March 1997). "RFC 2131 - Dynamic Host Configuration Protocol" (http://tools.ietf.org/html/rfc2131#section-7). *Network Working Group*. .

[9] Sergey Golovanov (Kaspersky Labs) (June 2011). "TDSS loader now got “legs”" (http://www.securelist.com/en/blog/208188095/TDSS_loader_now_got_legs). .

[10] Ted Lemon (April 2002). "Implementation of RFC 3118" (http://www.ietf.org/mail-archive/web/dhcwg/current/msg00876.html). .

References

External links

- RFC 2131 - Dynamic Host Configuration Protocol
- RFC 2132 - DHCP Options and BOOTP Vendor Extensions
- DHCP RFC (http://www.bind9.net/rfc-dhcp) - Dynamic Host Configuration Protocol RFCs (IETF)
- RFC 4242 - Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6
- DHCP Sequence Diagram (http://www.eventhelix.com/RealtimeMantra/Networking/DHCP.pdf) - This sequence diagram covers several scenarios of DHCP operation.
- RFC 3046 - DHCP Relay Agent Information Option
- Recommended Operation for Switches Running Relay Agent and Option 82 (http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB0088R0_ODVA_DHCP_Option_82v2.pdf) - Provides a description of how DHCP option 82 works.
- RFC 3942 - Reclassifying Dynamic Host Configuration Protocol Version Four (DHCPv4) Options
- RFC 4361 - Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)
- RFC 4436 - Detecting Network Attachment in IPv4 (DNAv4)
- DHCP Protocol Messages (http://support.microsoft.com/kb/169289) - A good description of the individual DHCP protocol messages.
- ISC DHCP (https://www.isc.org/software/dhcp) - Internet Services Consortium's open source DHCP implementation.
- Dynamic Host Configuration Protocol (DHCP) (http://www.microsoft.com/dhcp) - Microsoft TechNet
- Interactive DHCP basic simulation (http://www.learningocean.com/view.php?cid=1026&protocol=DHCP&title=1.DHCP Basics&ctype=1)

Hypertext Transfer Protocol

HTTP
Persistence · Compression · HTTPS
Request methods
OPTIONS · GET · HEAD · POST · PUT · DELETE · TRACE · CONNECT
Header fields
Cookie · ETag · Location · Referer
DNT · X-Forwarded-For
Status codes
301 Moved permanently
302 Found
303 See Other
403 Forbidden
404 Not Found

The **Hypertext Transfer Protocol** (**HTTP**) is a application protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web.

The standards development of HTTP has been coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs), most notably RFC 2616 (June 1999), which defines HTTP/1.1, the version of HTTP in common use.

Technical overview

HTTP functions as a request-response protocol in the client-server computing model. In HTTP, a web browser, for example, acts as a *client*, while an application running on a computer hosting a web site functions as a *server*. The client submits an HTTP *request* message to the server. The server, which stores content, or provides *resources*, such as HTML files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

A web browser (or client) is often referred to as a *user agent* (UA). Other user agents can include the indexing software used by search providers, known as web crawlers, or variations of the web browser such as voice browsers, which present an interactive voice user interface.

The HTTP protocol is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of the original, so-called *origin server*, to improve response time. HTTP proxy servers at network boundaries facilitate communication when clients without a globally routable address are located in private networks by relaying the requests and responses between clients and servers.

HTTP is an Application Layer protocol designed within the framework of the Internet Protocol Suite. The protocol definitions presume a reliable Transport Layer protocol for host-to-host data transfer.[2] The Transmission Control Protocol (TCP) is the dominant protocol in use for this purpose. However, HTTP has found application even with unreliable protocols, such as the User Datagram Protocol (UDP) in methods such as the Simple Service Discovery Protocol (SSDP).

HTTP Resources are identified and located on the network by Uniform Resource Identifiers (URIs)—or, more specifically, Uniform Resource Locators (URLs)—using the `http` or `https` URI schemes. URIs and the Hypertext Markup Language (HTML), form a system of inter-linked resources, called hypertext documents, on the Internet, that led to the establishment of the World Wide Web in 1990 by English computer scientist and innovator Tim Berners-Lee.

The original version of HTTP (HTTP/1.0) was revised in HTTP/1.1. HTTP/1.0 uses a separate connection to the same server for every request-response transaction, while HTTP/1.1 can reuse a connection multiple times, to download, for instance, images for a just delivered page. Hence HTTP/1.1 communications experience less latency as the establishment of TCP connections presents considerable overhead.

History

The term HyperText was coined by Ted Nelson who in turn was inspired by Vannevar Bush's microfilm-based "memex". Tim Berners-Lee first proposed the "WorldWideWeb" project — now known as the World Wide Web. Berners-Lee and his team are credited with inventing the original HTTP protocol along with the HTML and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server.[3] The response from the server was always an HTML page.[4]

The first documented version of HTTP was **HTTP V0.9** [5] (1991). Dave Raggett led the HTTP Working Group (HTTP WG) in 1995 and wanted to expand the protocol extended operations, extended negotiation, richer meta-information, tied with a security protocol and got more efficient by adding additional methods and header fields.[6] [7] RFC 1945 officially introduced and recognized HTTP V1.0 in 1996.

The HTTP WG planned to publish new standards in December 1995[8] and the support for pre-standard HTTP/1.1 based on the then developing RFC 2068 (called HTTP-NG) was rapidly adopted by the major browser developers in early 1996. By March 1996, pre-standard HTTP/1.1 was supported in Arena,[9] Netscape 2.0,[9] Netscape Navigator Gold 2.01,[9] Mosaic 2.7, Lynx 2.5, and in Internet Explorer 3.0. End-user adoption of the new browsers was rapid. In March 1996, one web hosting company reported that over 40% of browsers in use on the Internet were HTTP 1.1 compliant. That same web hosting company reported that by June 1996, 65% of all browsers accessing their servers were HTTP/1.1 compliant.[10] The HTTP/1.1 standard as defined in RFC 2068 was officially released in January 1997. Improvements and updates to the HTTP/1.1 standard were released under RFC 2616 in June 1999.

HTTP session

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80; see List of TCP and UDP port numbers). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested resource, an error message, or some other information.[1]

Request message

The request message consists of the following:

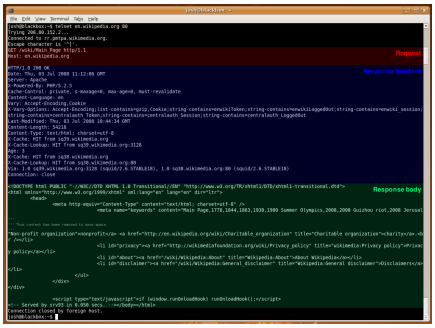
- Request line, such as `GET /images/logo.png HTTP/1.1`, which requests a resource called `/images/logo.png` from server
- Headers, such as `Accept-Language: en`
- An empty line.
- An optional message body.

The request line and headers must all end with <CR><LF> (that is, a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace.[11] Although <CR><LF> is required <LF> alone is also accepted by most servers. In the HTTP/1.1 protocol, all headers except Host are optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 specification in RFC1945 [12] [13].

Request methods

HTTP defines nine methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified **resource**. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.



An HTTP request made using telnet. The request, response headers and response body are highlighted.

HEAD

Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

GET

Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval".[1] The W3C has published guidance principles on this distinction, saying, "Web application design should be informed by the above principles, but also by the relevant limitations."[14] See safe methods below.

POST

Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

PUT

Uploads a representation of the specified resource.

DELETE

Deletes the specified resource.

TRACE

Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS

Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT

Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.[15]

PATCH

Is used to apply partial modifications to a resource.[16]

HTTP servers are required to implement at least the GET and HEAD methods[17] and, whenever possible, also the OPTIONS method.

Safe methods

Some methods (for example, HEAD, GET, OPTIONS and TRACE) are defined as *safe*, which means they are intended only for information retrieval and should not change the state of the server. In other words, they should not have side effects, beyond relatively harmless effects such as logging, caching, the serving of banner advertisements or incrementing a web counter. Making arbitrary GET requests without regard to the context of the application's state should therefore be considered safe.

By contrast, methods such as POST, PUT and DELETE are intended for actions that may cause side effects either on the server, or external side effects such as financial transactions or transmission of email. Such methods are therefore not usually used by conforming web robots or web crawlers; some that do not conform tend to make requests without regard to context or consequences.

Despite the prescribed safety of *GET* requests, in practice their handling by the server is not technically limited in any way. Therefore, careless or deliberate programming can cause non-trivial changes on the server. This is discouraged, because it can cause problems for Web caching, search engines and other automated agents, which can make unintended changes on the server.

Furthermore, methods such as TRACE, TRACK and DEBUG are considered potentially 'unsafe' by some security professionals, because they can be used by attackers to gather information or bypass security controls during attacks. Security software tools such as Tenable Nessus and Microsoft URLScan report on the presence of these methods as being security issues.

Idempotent methods and web applications

Methods PUT and DELETE are defined to be idempotent, meaning that multiple identical requests should have the same effect as a single request. Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol.[1]

In contrast, the POST method is not necessarily idempotent, and therefore sending an identical POST request multiple times may further affect state or cause further side effects (such as financial transactions). In some cases this may be desirable, but in other cases this could be due to an accident, such as when a user does not realize that their action will result in sending another request, or they did not receive adequate feedback that their first request was successful. While web browsers may show alert dialog boxes to warn users in some cases where reloading a page may re-submit a POST request, it is generally up to the web application to handle cases where a POST request should not be submitted more than once.

Note that whether a method is idempotent is not enforced by the protocol or web server. It is perfectly possible to write a web application in which (for example) a database insert or other non-idempotent action is triggered by a GET or other request. Ignoring this recommendation, however, may result in undesirable consequences, if a user agent assumes that repeating the same request is safe when it isn't.

Status codes

In HTTP/1.0 and since, the first line of the HTTP response is called the *status line* and includes a numeric *status code* (such as "404") and a textual *reason phrase* (such as "Not Found"). The way the user agent handles the response primarily depends on the code and secondarily on the response headers. Custom status codes can be used since, if the user agent encounters a code it does not recognize, it can use the first digit of the code to determine the general class of the response.[18]

Also, the standard *reason phrases* are only recommendations and can be replaced with "local equivalents" at the web developer's discretion. If the status code indicated a problem, the user agent might display the *reason phrase* to the user to provide further information about the nature of the problem. The standard also allows the user agent to attempt to interpret the *reason phrase*, though this might be unwise since the standard explicitly specifies that status codes are machine-readable and *reason phrases* are human-readable.

Persistent connections

In HTTP/0.9 and 1.0, the connection is closed after a single request/response pair. In HTTP/1.1 a keep-alive-mechanism was introduced, where a connection could be reused for more than one request.

Such *persistent connections* reduce request latency perceptibly, because the client does not need to re-negotiate the TCP connection after the first request has been sent.

Version 1.1 of the protocol made bandwidth optimization improvements to HTTP/1.0. For example, HTTP/1.1 introduced chunked transfer encoding to allow content on persistent connections to be streamed, rather than buffered. HTTP pipelining further reduces lag time, allowing clients to send multiple requests before a previous response has been received to the first one. Another improvement to the protocol was byte serving, which is when a server transmits just the portion of a resource explicitly requested by a client.

HTTP session state

HTTP is a stateless protocol. A stateless protocol does not require the server to retain information or status about each user for the duration of multiple requests. For example, when a web server is required to customize the content of a web page for a user, the web application may have to track the user's progress from page to page. A common solution is the use of HTTP cookies. Other methods include server side sessions, hidden variables (when the current page is a form), and URL-rewriting using URI-encoded parameters, e.g., `/index.php?session_id=some_unique_session_code`.

Secure HTTP

There are three methods of establishing a secure HTTP connection: HTTP Secure, Secure Hypertext Transfer Protocol and the HTTP/1.1 Upgrade header. Browser support for the latter two is, however, nearly non-existent, so HTTP Secure is the dominant method of establishing a secure HTTP connection.

Example session

Below is a sample conversation between an HTTP client and an HTTP server running on www.example.com, port 80.

Client request

```
GET /index.html HTTP/1.1□□
Host: www.example.com□□
□□
```

A client request (consisting in this case of the request line and only one header) is followed by a blank line, so that the request ends with a double newline, each in the form of a carriage return followed by a line feed. The "Host" header distinguishes between various DNS names sharing a single IP address, allowing name-based virtual hosting. While optional in HTTP/1.0, it is mandatory in HTTP/1.1.

Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

A server response is followed by a blank line and text of the requested page. The ETag (entity tag) header is used to determine if a cached version of the requested resource is identical to the current version of the resource on the server. *Content-Type* specifies the Internet media type of the data conveyed by the http message, while *Content-Length* indicates its length in bytes. The HTTP/1.1 webserver publishes its ability to respond to requests for certain byte ranges of the document by setting the header *Accept-Ranges: bytes*. This is useful, if the client needs to have only certain portions[19] of a resource sent by the server, which is called byte serving. When *Connection: close* is sent in a header, it means that the web server will close the TCP connection immediately after the transfer of this response.

Most of the header lines are optional. When *Content-Length* is missing the length is determined in other ways. Chunked transfer encoding uses a chunk size of 0 to mark the end of the content. *Identity* encoding without *Content-Length* reads content until the socket is closed.

A *Content-Encoding* like *gzip* can be used to compress the transmitted data.

References

[1] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee (June 1999). "RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1" (http://tools.ietf.org/html/rfc2616). .

[2] Fielding, et al. Internet RFC 2616." (http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html#sec1.4), section 1.4. Retrieved on January 21, 2009.

[3] Berners-Lee, Tim. "HyperText Transfer Protocol" (http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP.html). World Wide Web Consortium. . Retrieved 31 August 2010.

[4] Tim Berners-Lee. "The Original HTTP as defined in 1991" (http://www.w3.org/Protocols/HTTP/AsImplemented.html). World Wide Web Consortium. . Retrieved 24 July 2010.

[5] http://www.w3.org/pub/WWW/Protocols/HTTP/AsImplemented.html

[6] Raggett, Dave. "Dave Raggett's Bio" (http://www.w3.org/People/Raggett/profile.html). World Wide Web Consortium. . Retrieved 11 June 2010.

[7] Raggett, Dave; Berners-Lee, Tim. "Hypertext Transfer Protocol Working Group" (http://www.w3.org/Arena/webworld/httpwgcharter.html). World Wide Web Consortium. . Retrieved 29 September 2010.

[8] Raggett, Dave. "HTTP WG Plans" (http://www.w3.org/Arena/webworld/httpwgplans.html). World Wide Web Consortium. . Retrieved 29 September 2010.

[9] Simon Spero. "Progress on HTTP-NG" (http://www.w3.org/Protocols/HTTP-NG/http-ng-status.html). World Wide Web Consortium. . Retrieved 11 June 2010.

[10] "HTTP/1.1" (http://www.webcom.com/glossary/http1.1.shtml). *Webcom.com Glossary entry*. . Retrieved 2009-05-29.

[11] Cailliau, Robert (1 July 1992). "Updates To HTTP" (http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP/HTTP2.html). World Wide Web Consortium. . Retrieved 1 September 2010.

[12] http://www.ietf.org/rfc/rfc1945.txt

[13] "Apache Week. HTTP/1.1" (http://www.apacheweek.com/features/http11). . 090502 apacheweek.com
[14] Jacobs, Ian (2004). "URIs, Addressability, and the use of HTTP GET and POST" (http://www.w3.org/2001/tag/doc/whenToUseGet.html#checklist). *Technical Architecture Group finding*. W3C. . Retrieved 26 September 2010.
[15] "Vulnerability Note VU#150227: HTTP proxy default configurations allow arbitrary TCP connections" (http://www.kb.cert.org/vuls/id/150227). US-CERT. 2002-05-17. . Retrieved 2007-05-10.
[16] Dusseault, Lisa; Snell, James M.. "RFC 5789: PATCH Method for HTTP" (http://tools.ietf.org/html/rfc5789). .
[17] "HTTP 1.1 Section 5.1.1" (http://tools.ietf.org/html/rfc2616#section-5.1.1). Tools.ietf.org. . Retrieved 2010-08-01.
[18] "6.1 Status-Line" (http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1). W3.org. . Retrieved 2010-08-01.
[19] Tools.ietf.org (http://tools.ietf.org/html/draft-ietf-http-range-retrieval-00), Byte Range Retrieval Extension to HTTP

Further reading

- HTTP 0.9 - As Implemented in 1991 (http://www.w3.org/Protocols/HTTP/AsImplemented.html)

External links

- "Change History for HTTP" (http://www.w3.org/Protocols/History.html). W3.org. Retrieved 2010-08-01. A detailed technical history of HTTP.
- "Design Issues for HTTP" (http://www.w3.org/Protocols/DesignIssues.html). W3.org. Retrieved 2010-08-01. Design Issues by Berners-Lee when he was designing the protocol.
- "Classic HTTP Documents" (http://www.w3.org/Protocols/Classic.html). W3.org. 1998-05-14. Retrieved 2010-08-01. list of other classic documents recounting the early protocol history

Article Sources and Contributors

Communications protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=463428398 *Contributors*: 16@r, Ahoerstemeier, Aldie, Allan McInnes, Amillar, Ardonik, B4hand, BAxelrod, Bctwriter, Bert490, Bhamv, Bloodshedder, Bodachi, BodyTag, BonsaiViking, Brainmachine, Brian2wood, Byrial, Calle, Can't sleep, clown will eat me, CanadianLinuxUser, CapitalR, Charivari, Chowells, Christian75, Chuunen Baka, Ciampix, Coasterlover1994, Conti, Conversion script, Cremepuff222, DVdm, Dairyintheillawarra, Damian Yerrick, David Woodward, Dekisugi, Denisutku, Dgreen34, Dyl, Eddie Nixon, Edward, ElComandanteChe, Elmindreda, Enric Naval, EoGuy, Eugene-elgato, Eumolpo, Ferryca, Flowanda, Fumblebruschi, Gensanders, Glenn, Graham87, Guaka, Hairy Dude, Hannes Hirzel, Harryzilber, Heavenlyhash, Helland, Helmsjs, Hgilbert, Hwardsil, IlyaKorolev, Intelati, J.delanoy, JEG, JLaTondre, JTN, Jacen6788, Javlonson, Jennypei, Jim.henderson, Jnc, Jncraton, Joe102283, Jphastings, Jsoon eu, Kbdank71, Kbrose, Kevinalewis, Kku, KnowledgeOfSelf, Kvng, Kyz, Leafyplant, Leptictidium, Lerdsuwa, Liempt, LilHelpa, Lockley, Mange01, Materialscientist, Matthuxtable, Matusz, Maximaximax, Michael Hardy, MichaelJanich, Mild Bill Hiccup, MischaO, Miym, Moazami, Moogwrench, Mormegil, Mrpeterobrien, Mulad, Mushroom, Mwanner, Mxn, Nageh, Niteowlneils, Nixdorf, Orange Suede Sofa, Oscarthecat, Oyvind.misje, Pegasus1138, Petruss, Phatom87, Piet Delport, Pillaidee, R'n'B, Radagast83, RedWolf, Regin LARSEN, Rememberway, RexNL, Rich Farmbrough, Rick Block, Rick Sidwell, Riick, Rjstott, Robofish, Romangs, RyanCross, S.K., Sachipra, Salsal99, Samtheboy, ShaunMacPherson, SimonP, Skarg, Smalljim, Smallman12q, Snoyes, Sonja Connell, Tagishsimon, Takanoha, The Anome, The Rambling Man, Thierryc, Tirupur123451, Tristanb, Una Smith, Undsoft, Unyoyega, Vegetator, Velella, Vonkje, Vrenator, WhisperToMe, Whosasking, Wiki alf, Wtshymanski, X!, Yekrats, Ykhwong, Youandme, Yyy, Zipz0p, Zondor, 276 anonymous edits

Computer network *Source*: http://en.wikipedia.org/w/index.php?oldid=465443350 *Contributors*: *Kat*, 10metreh, 149AFK, 16@r, 1966batfan, 28421u2232nfenfcenc, 28bytes, 2D, 5 albert square, 802geek, A520, ACBest, AJCham, ARUNKUMAR P.R, Aajaja, Abb615, Abraham, B.S., Acnetj, Adam1213, Adam850, Adambro, Addihockey10, AdjustShift, AeonicOmega, Aeonx, Aicchalmers, Aij, Aitias, Ajl772, Akb.pcb, Akendall, Alansohn, Aleenf1, AlexiusHoratius, Alpha 4615, Alphachimp, Altruism, Alucard 16, Amirrad.en, Andrew D White, Andrewcrawford, Andy16666, Aneah, Angrysockhop, Anna Lincoln, Anonauthor, Anonymi, Antandrus, Anupkeskar, Apau98, Apparition11, Apteva, Arctic Fox, ArglebargleIV, Armchair info guy, ArmenSokhakyan, Arthena, Atw1996, Avinash0147, Avoided, Avril0412, AzaToth, Banaticus, Baronnet, Bart133, Bassbonerocks, Beelaj, Bella Swan, Belovedfreak, BevinBrett, Bhasrini, Bjankuloski06en, Black Falcon, Blanchardb, Blaxthos, Blazerskj7315, BlueDevil, Bluerasberry, Bob f it, Bobby122, Bobo192, Bogey97, Bokunenjin, Bongwarrior, Brandon, Brian Crawford, Btilm, BuddhaBubba, Bumpusjames, Bushsf, ButOnMethItIs, Bwmitche, Bxn1358, C777, CBM, CWY2190, CWii, Cachiadavid, Caiaffa, Calltech, Caltas, Camw, Can You Prove That You're Human, Can't sleep, clown will eat me, CanadianLinuxUser, CapitalR, Capricorn42, Captain panda, Captain-tucker, Captainreiss, CardinalDan, Cbdorsett, Cerebellum, CesarB, Cgmusselman, Chad44, Chakkalokesh, Chandlermbing, Chris the speller, Chrisch, Chriswiki, Chromaticity, Chzz, Clarince63, Closedmouth, Cnilep, Cnkids, Coeus559, CommonsDelinker, Corruptcopper, Courcelles, Cpiral, Cpl Syx, Cst17, Cstratacos, Cwilso, Cwoodskareska, Cxz111, Cybercobra, Cyclonenim, D, D. Recorder, D.c.camero, DARTH SIDIOUS 2, DJBullfish, Dabomb87, Dac04, DakotaDAllen, DaltinWentsworth, DanMS, Daniel Procter, Danlaycock, Dap263, Darkkloud29, Darth Panda, Davosmith, Dayyanb, DeadEyeArrow, Decltype, Demonslayer4000, DerHexer, Dgtsyb, Diannaa, Dicklyon, Discospinster, Djg2006, Djmckee1, Dlohcierekim, Dotancohen, DoubleBlue, Download, Dreadstar, Dwayne, DyingIce, Dylan620, EarthPerson, Easwarno1, Eeekster, Eggnogicecream, Egmontaz, Eitheladar, Eleos, Elinruby, Emailtonaved, EncMstr, Enigmaman, Epbr123, Er Komandante, Eraxx, Erdoğan said, Erianna, Eric-Wester, Escape Orbit, EvokeNZ, Excirial, Extransit, FJPB, Face, Faithlessthewonderboy, Falcon8765, Faradayplank, Favonian, FayssalF, Fazlurrahman95, Fgrose, Fieldday-sunday, Fiftwekid, Figma, Finnysgay, Fintanmurphy, Fireaxe888, Fireice, Firien, Flewis, Flyingcheese, Fozz79, Frankie0607, Freedomlinux, Frmatt, Funnyfarmofdoom, Fyyer, Fæ, GLaDOS, GNMC, Gail, GandalfDaGraay, Gholam, Giftlite, Glane23, Glenn, Gnowor, Gogo Dodo, GoingBatty, Gonchibolso12, GoneAwayNowAndRetired, Grafen, Greenrd, GroveGuy, Guggfe, Gurch, Gurchzilla, Gyan pokhara, Hammersoft, Happysailor, HarisM, Harryzilber, Harshrateria, Haseo9999, Hcberkowitz, Hchrm, Headbomb, Hellokittylover12, Hemachandra18, Herbythyme, HiDrNick, Hpcanswers, Hut 8.5, INkubusse, Ianmillner, Igoldste, Imroy, Indon, Inseeisyou, Intgr, Ipatrol, Iridescent, IronGargoyle, It4it-wiki, Itusg15q4user, J. Martin Wills, J.delanoy, JBazuzi, JEBrown87544, JLRedperson, JLaTondre, Jackelfive, Jackfork, Jackol, Jake Wartenberg, Jan1nad, Japheth the Warlock, Jared Preston, Jasper Deng, Jauerback, Jay, Jayakrishnan0804, Jclemens, Jeepday, Jeff G., Jer71, Jerzy, Jflabourdette, Jheyahr, Jimmi Hugh, Jjasi, Jjensen347, Jjron, Jni, Joantorres, Joelhanley, John Stumbles, Johnuniq, JonHarder, Jordanfeldman, Joseph Solis in Australia, Joshua Gyamfi, Joshua Scott, Joy, Jpbowen, Jxl180, KGasso, Kazochi, Kbrose, Keilana, KelleyCook, Kgfleischmann, Kieferb12, King of Hearts, Kingpin13, Kms, KnowBuddy, Knowz, Koffieyahoo, Koolabsol, Kozuch, Kranix, Krishnavedala, Krj373, Kudret abi, Kvng, L Kensington, L314t, L33th4x0rguy, LFaraone, Landon1980, Leafyplant, LeaveSleaves, LeinaD natipaC, Levineps, Leye1, Lightmouse, Lights, LilHelpa, Lilac Soul, Lillightning, Limideen, LindsayH, Ling.Nut, Loile0801, Lotje, Lucky arien2001, Ludovic.ferre, Luk, Lukeritchie, MER-C, Macy, Madhero88, Magioladitis, Maismuhanad, Mandarax, Maneendra, Manjax76, Marco94, Marek69, MarkBolton, Markdude09, MarkmacVSS, Mascharanas, Masterjamie, Matdrodes, Mattgirling, Mattl2001, McSly, Mdd, Mentifisto, Mephistophelian, Metricopolus, Michael Angelkovich, Michael93555, Micke-sv, Mikaey, Mike.lifeguard, MikhailVS, Milind m2255, Mindmatrix, Minimac, Minimac's Clone, Misortie, MisterCharlie, Miym, Mlewis000, Mlouns, Mmernex, Mmmeg, Modamoda, Montchav, Montrevux, MorrisRob, Mottsauce, Mr. Wheely Guy, MrSmook, Mrnatural, MuZemike, Muhandes, My76Strat, Mynameiswill, N5iln, Nanzilla, Nathanid, NawlinWiki, Nazi 2007, Neillucas, NeoJustin, Nepenthes, Netalarm, Nethgirb, Netito777, Networkingguy, NewEnglandYankee, Ngriffeth, Nihiltres, Nopetro, Northamerica1000, NorwegianBlue, NoticeBored, Novalis, Nrm123, Nubiatech, Nurg, Nurlan926, Ojasweesharma, Oliver202, OllieFury, Onure, Opelio, Orange Suede Sofa, Orbst, Oroso, Otolemur crassicaudatus, OverlordQ, Oxymoron83, PAntoni, Padillah, Pakakj.20june, Pascal.Tesson, PatrikR, Paul August, Pchov, Pdcook, Pedro, Peterwhy, PhJ, Pharaoh of the Wizards, Phatom87, PhilKnight, Philip Trueman, PhilipJS, Phoe6, Piano non troppo, Pigman, Pill, Pinethicket, Pingveno, Plaga701, PleaseStand, Polluxian, Porkrind, Porterjoh, Possum, Poweroid, Prari, Prashanthns, Promethean, Public Menace, Puffin, Pwarrior, Pxma, Pyrospirit, Quaeler, Quantpole, Qwyrxian, Qxz, R'n'B, RJaguar3, RadioFan2 (usurped), Radon210, Raed abu farha, Rahul440, RainbowOfLight, Rananera74, Rangek, Rangoon11, Raven21421, Razarajpoot, Razertek, Rctay, Reaper Eternal, Red Thrush, Reliableforever, Rememberway, Renewer, Rettetast, Riana, Ricardoprojects, Rick Block, Riick, Rmaheshnaidu, Rmosler2100, RoMo37, RobertIllston, Roland Kaufmann, Rpsjrcpa, Rsrikanth05, Rwwww, RyanCross, Ryulong, SMC, ST47, Saketmitra, Sandeepsp4u, Satori Son, Sayedomer, Scgtrp, Sciurinæ, Seaphoto, Seek54, Sephiroth BCR, Sesu Prime, Sgeo, Shadowjams, Shanes, Shenme, Shiwakant.bharti, ShornAssociates, Sigma 7, Skarebo, SkyWalker, Skyezx, Slon02, SmartGuy, Smokizzy, Snottywong, SoCalSuperEagle, Soap Poisoning, Some jerk on the Internet, Sonjaaa, SpaceFlight89, SpecMode, Special Cases, Spitfire, Spmeyn, SpuriousQ, Ssmorris, StaleOnion, SteinbDJ, Stephan Leeds, Stephenb, Storm Rider, Stwalkerster, Suffusion of Yellow, Sumitprksh, SuperSlacker, Symetrix, Synchronism, Syrthiss, Tango, Tarif Ezaz, Tbhotch, TechTWI, Technobadger, Terrek, Testbells, Tgeairn, The Thing That Should Not Be, TheCatalyst31, Thedjatclubrock, Thejokerface, Thetaung, Think outside the box, Thrindel, Thumperward, Tiddly Tom, Tide rolls, Titoxd, Todd Peng, Tombomp, Tony1, Topbanana, Toussaint, Traxs7, Trenwith, TreveX, Trgaz, Triona, Triwbe, Tslocum, Turabsf, Turgan, TutterMouse, Tyrol5, UBJ 43X, UU, Ultimarko, Uncle Dick, UncleDouggie, Vamphemu, VandalCruncher, Vanished user 39948282, Vanka5, Versageek, Versus22, VictorAnyakin, Visaforu, Visor, VooDooChild, Voyagerfan5761, Vrenator, W Nowicki, Wa3frp, Wadamja, Walter.bender, Waryklingon, Watchdog9, Weezey, Welsh, Whereizben, WhisperToMe, WikHead, Wikipelli, Wikipixel, Wilde Jagd, Wimt, Wine Guy, Wizardist, Wknight94, Wolfkeeper, Woohookitty, Wtmitchell, Wtsao, Wuhwuzdat, Xompanthy, YassineMrabet, ZamorakO o, Zawthet, Zedex7, Zentraleinheite, Zzuuzz, מבעת-זרם, ترجمان05, محبوب عالم, شات صوتي, 2716 anonymous edits

OSI model *Source*: http://en.wikipedia.org/w/index.php?oldid=465441964 *Contributors*: 0612, 0x6D667061, 1337 JN, 24.12.199.xxx, 28bytes, 63.227.96.xxx, 7, 75th Trombone, 802geek, @modi, A930913, ABF, AK Auto, Abarry, Abune, Adamantios, Addshore, Adibob, Adityagaur 7, Adj08, Adoniscik, Adrianwn, Advancedtelcotv, Ageekgal, Ahoerstemeier, Aitias, Ajo Mama, Ajw901, Alansohn, Albanaco, Aldie, Ale jrb, AlistairMcMillan, Alphachimp, Alucard 16, Alvestrand, Amillar, Amitbhatia76, Amtanoli, Andjohn2000, Andre Engels, Andybryant, Angrysockhop, Animum, Anjola, Anna Lincoln, Anonymous anonymous, Another-anomaly, Apocryphite, Apparition11, Arroww, Artur Perwenis, Arunachalammanohar, Ashutosh.mcse, Aslambasha09, Asn1tlv, AtomicDragon, Atreyu42, Audunv, Avitesh, AxelBoldt, Ayengar, B4hand, BACbKA, BDerrly, Bakilas, Balajia82, Bariswheel, Bchiap, Bdamokos, Beelaj, BenLiyanage, Beno1000, Biblbroks, Bjelleklang, Bletch, Blueskies238, Bmylez, Bobo192, Bogdangiusca, Boikej, Bojer, BommelDing, Bonobosarenicer, Booyabazooka, Borgx, Brambleclawx, Brandon, Brick Thrower, Brougham96, Bryan Derksen, BuickCenturyDriver, Bzimage.it, Bücherwürmlein, CDima, CIreland, CMBJ, Caerwine, Caesura, Calmer Waters, Caltas, CambridgeBayWeather, Camw, Can You Prove That You're Human, Can't sleep, clown will eat me, CanadianLinuxUser, Candc4, Caper13, Carre, Casey Abell, Causa sui, Cburnett, Cbustapeck, Cflm001, Charles Edward, Charm, Che090572, Chester Markel, Chfalcao, Chimpex, ChiragPatnaik, Chrislk02, Chupon, Citicat, Closedmouth, Cokoli, Cometstyles, Conquest ace, Conversion script, Coriron, Courcelles, Cputrdoc, CraSH, CraigBox, Crasheral, Crimsonmargarine, Cs mat3, Ctbolt, Cxxl, Cybercobra, CyborgTosser, Cyktsui, CynicalMe, DARTH SIDIOUS 2, DJPohly, DSParillo, Damian Yerrick, Daniel, Danlev, Dave2, David Edgar, David Gerard, David0811, DavidBarak, DavidLevinson, Davidjk, Dcooper, Dcovell, Deagle AP, Delfeye, Delldot, DeltaQuad, Demitsu, DennyColt, Dgtsyb, Dicklyon, Dili, Dino.korah, Discospinster, Dispenser, Djib, Djmoa, DmitryKo, Dmohantyatgmail, Doniago, Dpark, DrDOS, DrSpice, Drat, Dreish, Drwarpmind, Duey111, Dumbledad, Dzubint, EJDyksen, EJSawyer, ENeville, EagleOne, Eazy007, Ed g2s, EdH, Edivorce, Edward, ElKevbo, Eldiablo, Eleassar, Elfosardo, Eliezerb, Elipongo, Emperorbma, EnOreg, Enjoi4586, Enochlau, Epbr123, Eric Soyke, Everyking, Evillawngnome, Ewlyahoocom, Excirial, FF2010, Falk.H.G., Fang Aili, Feezo, Fiable.biz, Filemon, Finlay McWalter, Fjpanna, Fleg31789, Flewis, Flowanda, FrankTobia, Fred Bradstadt, Fredrik, Free Bear, FreshPrinz, Fresheneesz, Friday, Friedo, Friginator, Fullstop, Fumitol, Fuzheado, Fvw, Fæ, GDonato, Gadfium, Gafex, GarethGilson, Gary King, Gasp01, Gazpacho, Geek2003, General Rommel, Ghostalker, Giftlite, Gilliam, GlassCobra, Glenn, Goodnightmush, Graeme Bartlett, Grafen, Graham.rellinger, GreYFoXGTi, Grendelkhan, Grubber, Gsl, Gurchzilla, Guy Harris, Gwernol, Gökhan, H2g2bob, H34d, HMGb, Haakon, Hadal, HamatoKameko, HarisM, Hatch68, Hcberkowitz, Hdante, Helix84, Hellomarius, Henrikholm, Herbee, Heron, Hes Nikke, Hetar, HexaChord, Hgerstung, Hiddekel, Highpriority, Honeyman, I dream of horses, IMSoP, IReceivedDeathThreats, Iambk, Ideoplex, Ifroggie, Ilario, Immunize, Inkhorn, Inkling, Insineratehymn, Intgr, Inversetime, InvisibleK, Iridescent, IronGargoyle, Ishikawa Minoru, Island Monkey, Isofox, Isthisthingworking, Itpastorn, Itusg15q4user, Iviney, J.delanoy, JMatthews, JV Smithy, Jake Wartenberg, JamesBWatson, Jannetta, Jatinsinha, Jauerback, Jchristn, Jcw69, Jdrrmk, JeTataMe, Jeanjour, Jeff G., Jeffrey Mall, Jessemerriman, Jetekus, Jhilving, JidGom, Jjenkins5123, Jmorgan, Jnc, JoanneB, JodyB, Joebeone, John Hopley, John Vandenberg, John254, Johnblade, Johnleemk, Johnuniq, JonHarder, Jonathanwagner, Jonwatson, Joodas, Josef Sábl cz, Josh Parris, Jovianeye, Joy, Jpta, Jrodor, Jschoon4, Jusdafax, Kaaveh Ahangar, Kallaspriit, Karelklic, Karpouzi, Kaszeta, Katalaveno, Kaz219, Kazrak, Kbrose, Kcordina, KerryVeenstra, Kesla, Kevin Rector, Kgrr, Khat17, Killiondude, Kim Rubin, Kingpin13, Kirill Lokshin, Kkbairi, KnowledgeOfSelf, Kraftlos, Kramerino, Krampo, Krellis, Kuru, Kvng, Kyllys, LOL, LOTRrules, Lachlancooper, Lankiveil, Lawrence Cohen, Lazarus666, Leafyplant, Lear's Fool, Lectonar, Lee Carre, Lights, LittleOldMe, LizardJr8, Lockcole, Logictheo, Logthis, Lomn, Looxix, Lord Chamberlain, the Renowned, Lordeaswar, Lotje, Lulu of the Lotus-Eaters, Luna Santin, Lupin, Lynnallendaly, M, MBisanz, MER-C, MIT Trekkie, Maguscrowley, Mahanga, Mahesh Sarmalkar, Majorly, Mange01, Manishar us, Marek69, MarkSutton, MarkWahl, Markb, Markhurd, Markolinsky, MartinHarper, Martinkop, Marvin01, Materialscientist, Mattalyst, Matthew Yeager, Mattjgalloway, Mattmill30, Mbc362, Mboverload, McGinnis, Mcnuttj, Mdd, Meepster, Mendel, Mephistophelian, Merlion444, Metaclassing, Micahcowan, Michael Hardy, Michael miceli, Mike Rosoft, Mikel Ward, Mikeo, Mikeyh56, Milind m2255, Minimac, Mkweise, Mlewis000, Mmeerman, Mmernex, Mmmeg, Mobius R, Mohitjoshi999, Mohitsport, Mojalefa247, Monterey Bay, Morten, Moxfyre, Mr Elmo, Mr Stephen, Mr.ghlban, MrOllie, Mrankur, MrsValdry, Mtd2006, MuZemike, Mudasir011, Mulad, Mwtoews, Myanw,

Myheadspinsincircles, N-Man, N5iln, Naishadh, Nanshu, Naohiro19, Naresh jangra, Nasa-verve, Natarajuab, Nate Silva, Nathanashleywild, NawlinWiki, Nbarth, Nbhatla, Nejko, Nemesis of Reason, Nethgirb, Netsnipe, Niaz, Nick, Nickshanks, Nicolas1981, Nisavid, Nitecruzr, Nivix, Nk, Noahspurrier, Nolyann, Nsaa, Nubiatech, NuclearWarfare, Nux, OSUKid7, Odie5533, Ogress, Oita2001, OlEnglish, Omicronpersei8, Orange Suede Sofa, Originalharry, Ott, Ottosmo, Ouishoebean, Oxymoron83, PGWG, Palltrast, Pamri, Panser Born, Paparodo, Parakalo, Pastore Italy, Patch1103, Patrikor, Patstuart, Paul August, PaulWIKIJeffery, Pb30, Peni, Penno, Pethr, Petrb, Phatom87, Phil Boswell, Philip Trueman, PhilipMW, Pluyo8989, Pmorkert, Pointillist, Postdlf, Postmortemjapan, Praggu, ProPuke, Pseudomonas, Psiphiorg, Public Menace, Puchiko, Puckly, PyreneesJIM, Pytom, RainbowOfLight, Raju5134, RandomAct, Ravikiran r, RazorICE, Rcannon100, Rebroad, Recognizance, RedWolf, Reedy, Rejax, Rettetast, Rfc1394, Rgilchrist, Rhobite, Rich Farmbrough, RichardVeryard, Richwales, Rick Sidwell, Rjgodoy, Rjstinyc, Rlaager, Rnbc, RobEby, Robert K S, RobertL30, RockMFR, Rohwigan03, Ronz, Roo314159, RoscoMck, RossPatterson, Roux, Roux-HG, RoyBoy, Rsiddharth, Runis57, Runtux, Rursus, Ryan au, Ryt, Ryulong, S, S3000, SMC, Saad ziyad, Saddy Dumpington, Safety Cap, Saintfiends, Sakurambo, Savh, SaxicolousOne, Scarian, Schumi555, Scientus, Scohoust, Scolobb, Scottonsocks, Seaphoto, Sesu Prime, Shadow1, Shadowjams, SharePointStacy, Shell Kinney, Shirik, Shoeofdeath, ShornAssociates, Shraddha deshmukh, Shrofami, Sietse Snel, Simonfl, Simple Bob, SineChristoNon, Sir Nicholas de Mimsy-Porpington, Skier Dude, Sliceofmiami, Slrobertson, Smalljim, Smokizzy, SnowFire, Snowolf, Soosed, Sp33dyphil, SpaceFlight89, Speaker to Lampposts, SpeedyGonsales, Spitfire8520, SpuriousQ, Sridev, StaticGull, Stemonitis, Stephan Leeds, Stephen Gilbert, StephenFalken, Stevage, Steven Zhang, StuartBrady, Subfrowns, Sunilmalik1107, Suruena, Swapcouch, Syntaxsystem, TAS, THEN WHO WAS PHONE?, Tagishsimon, Tangotango, Tarekradi, Tbsdy lives, Tcncv, Techtoucian, Tedickey, Tellyaddict, Tempodivalse, The Anome, The Athlon Duster, The Haunted Angel, The Thing That Should Not Be, Therumakna, Thief12, Thingg, Think4amit, ThreeDee912, ThunderBird, Tide rolls, Tim Q. Wells, TinyTimZamboni, Tom harrison, TomPhil, Tommy2010, Tompsci, Tony1, Tooki, Tpbradbury, Tpvibes, Tranzz, Travelbird, Tree Biting Conspiracy, Trevor MacInnis, Triona, TripleF, Triwbe, Troy 07, Turb0chrg, Tyler.szabo, UU, Umair ahmed123, Uncle Dick, Unkownkid123, Venu62, Versus22, VidGa, Vishnava, Visor, Vk anantha, Vmguruprasath, Voidxor, WLU, Waggers, Warrierrakesh, Wayfarer, Weregerbil, Whitejay251, WikiDan61, Wikipelli, William Avery, Willking1979, Wilson.canadian, Wily duck, Wingman4l7, Wire323, Wireless friend, Wishingtown, Wknight94, WoiKiCK, Woohookitty, Wrlee, Wrs1864, Wtmitchell, Wtshymanski, Yamamoto Ichiro, YamiKaitou, Yamike, Yms, YolanCh, Yuokool12, ZX81, ZachPruckowski, Zachary, Zoobee79, Žmogus, 3106 anonymous edits

TCP/IP model *Source*: http://en.wikipedia.org/w/index.php?oldid=464496019 *Contributors*: 5 albert square, Abdull, Acprisip, Alexandria, Altenmann, Amarikar, Amire80, Anon lynx, Atrizu, B3virq3b, Banshee Dauntle, Blurble, Bryan Derksen, CaseyPenk, Cashthischeck, Cburnett, Chicago god, Chimpex, Ciotog, Closedmouth, Cosmo0, Cybercobra, Davidelliottbell, Dgtsyb, Dthomsen8, EagleOne, Edgarde, Epheterson, Ewger, F00kie, Fred Gandt, Fresheneesz, Froissart, GFellows, GTBacchus, Gambler132, Giuliopp, Gogo Dodo, Gracenotes, Gscshoyru, Guy Harris, Hagaland, Hcberkowitz, I am Me true, IMSoP, Imroy, Indeterminate, Ino5hiro, Jmkelly, Jnc, John Vandenberg, Johnuniq, Karl Dickman, Kbrose, Kgfleischmann, Kkkdddiii, KnowledgeOfSelf, Kvng, L7HOMAS, Lemonflash, Lightdarkness, Lwood1281, MER-C, Magioladitis, Mandarax, Mange01, Markb, Marneus90, Martin.Budden, Materialscientist, Mebden, Metaclassing, Methecooldude, Mmernex, Mulad, Mwtoews, NerdyScienceDude, NotAnonymous0, Nubiatech, Obtext, Opelio, Orange Suede Sofa, Penwhale, Petrb, Phatom87, Pingveno, Pkoistin, PoisonedQuill, Porkrind, Prunesqualer, Public Menace, R'n'B, Rabbitfang, Raysonho, Rbmcnutt, Rednblu, RichardVeryard, Rick Sidwell, Robert K S, SF007, Samdutton, SaxicolousOne, Seaphoto, Shoessss, Siddhant, Speagles, Spellmaster, Squideshi, Stephan Leeds, Strainu, Strangerer, SymlynX, TN55, Tericee, ThFabba, ThinkEnemies, Tide rolls, Travellingbarrett, Tuxa, Valery.vv, W163, Wik, Wknight94, Woohookitty, Wrogiest, Wrs1864, Wtmitchell, Xcslacker100, Zac439, Zahra hashemi, Zvar, Тиверополник, 253 anonymous edits

Address Resolution Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=460784206 *Contributors*: 0x55534C, Abdull, Aldie, Alfio, Alvin-cs, Andareed, Arkrishna, Badams5115, Bcollins92131, Bernfarr, BillySharps, Bmaisonnier, Borgx, CarpeCerevisi, Casey Abell, Cburnett, CecilWard, Chealer, Cv-natalay, Dizon Ingens, Dkhydema, Dlohcierekim's sock, Dmccarty, DonDiego, Dorins, Doug.otis, Dysprosia, Edward, Edward Z. Yang, Eeekster, Enjoi4586, Enoent, Etienne.navarro, Euchiasmus, Fireman biff, Fred Gandt, Fredrik, Gamera2, Gerixau, Giftlite, Glenn, Grawity, Guenthert, Guy Harris, Hkulkarni, Hulkster121382, Huwr, IanHarvey, Ibc111, Isnow, JTN, Jeltz, JesseW, Jim-j, Johncc330, Johnuniq, Jqgarcia, Karada, Karih, Karl-Henner, Kbrose, Kelly Martin, Kenyon, Kjj31337, Kku, Kondareddy, Kranthiswaroop, Ksn, Kvng, Leftturnsolutions, Lilac Soul, Lollerskates, Looxix, M. B., Jr., M4gnum0n, MER-C, Mandarax, Mange01, MeekMark, MessiFCB, Mindmatrix, Misterfinsmiles, Ml198, Mokhov, Mouse Nightshirt, Mr Stephen, Muhandes, Music Sorter, Nageh, Nate Silva, Natersoz, Nealcardwell, Nixdorf, Nubiatech, Nuno Tavares, Olivier Debre, OsamaBinLogin, Papadopa, Pit, PolarYukon, QmunkE, Quarl, RB972, Rabarberski, Radiant chains, Rafostry, Raptor66, Rich Farmbrough, Richi, Rkinder, RobertKennedy, Roberta F., Rs junior, S4m1, SQL, SensuiShinobu1234, Sfisher, Sho Uemura, Spikey it, Stephan Leeds, Stormie, Suisui, Suruena, Swlee, Synergy, Tedernst, Template namespace initialisation script, The Anome, Tizio, Ttwaring, Uruiamme, Vanished 6551232, Voidxor, Wbenton, Wk muriithi, Woohookitty, Yurik, Yyy, Zundark, 318 anonymous edits

Asynchronous Transfer Mode *Source*: http://en.wikipedia.org/w/index.php?oldid=464394112 *Contributors*: *drew, Adashiel, Adriao, Alansohn, Aldie, Alex, Allstarecho, AnonMoos, Anthony Appleyard, Antti29, Anuragkothari, Apparition11, Arunachalammanohar, Ashok.Nair, Atm153, Azerekh, Backslash Forwardslash, Barek, Beaumont, Beej, BernardM, Betacommand, Blaxthos, Bluemoose, Bluezy, Bobblewik, Bobo192, Bobstopper, Brentdax, Brholden, Camw, Carbonlethagy, Cburnett, Chaitanya.lala, Cheesysam, Chenzw, Chester Markel, Chrisc, Chrischambers, Chrylis, Colin Marquardt, Conversion script, Creidieki, Cwolfsheep, DARTH SIDIOUS 2, Daniel Luechtefeld, Darrell Greenwood, Darrien, DataWraith, Davidoff, Demonkoryu, Derek Ross, Dgepstein, Dgtsyb, Dicklyon, Dinomite, Dr Debug, Drpixie, Duraivelanc, Dyl, E2eamon, EagleOne, Edaelon, Edokter, Edward, ElKevbo, Electron9, Emiller213, Enduser, Ericjmonson, Espoo, Everyking, Floppytoes, Frankie0607, Gaius Cornelius, Genghiskhanviet, Gerry Ashton, Get It, Giftlite, Gr33ndata, Graham.Fountain, Graham87, Gwicke, Harris7, Het, Hyad, Hydrowire, Iamzemasterraf, Ilario, Ino5hiro, IoannisK, Iridescent, Ivansanchez, Janto, Jec, Jesse Viviano, Jim.henderson, Jivecat, Jkl, Jnc, Joconnor, JohnGrantNineTiles, Jojalozzo, Jsung123, Karl-Henner, Kbrose, Kdaley, Keilana, Kelly Martin, Ketil3, Kevintolly, Kiffahh, KjellG, Kjkolb, Kozuch, Kukini, Kvng, Kylebrz, LachlanA, Leandrod, Lerdsuwa, Les boys, Lord Chamberlain, the Renowned, Ls6777, M412k, Mange01, MarkWahl, Markrod, Martarius, Martensjd, Materialscientist, Matt Crypto, Melsaran, Methossant, Michael Hardy, Mmkh, Mohanchander, Monedula, MonoAV, Mosfet RJF, Mzabaluev, Naohiro19, Neilrieck, Nezzadar, NickBush24, Niteowlneils, Nubiatech, Nurg, Oblivious, Ohnoitsjamie, Oli Filth, Olof nord, Onco p53, Ozzie13, Panda34, Paul Koning, Pedant17, Pembers, Pengu, Phatom87, Phil Holmes, Philip Trueman, Philippe, Piet Delport, Pleyden, Populus, Project2501a, Propeng, Public Menace, Quarl, RHaworth, Ra6907, RadioFan, RedWolf, Requestion, RexNL, Rich Farmbrough, Richard cocks, Richfife, Riffle, Rjamorim, Rl, Rookkey, Rrburke, RxS, Ryan Norton, Sakus, Saxbryn, Silent SAM, Skintigh, Sladen, Smfavaro, Smyth, Snowolf, Soumyasch, Spearhead, Sricciar, Stickee, Sumankane, TFriesen, Tagishsimon, Tanketz, Tannerbrockwell, Tekdude, Tetzcatlipoca, The Anome, Thebeginning, Timberframe, Tobias Hoevekamp, Tomg11030, Tommy2010, Totel, TripleF, Vaibhav.nimbalkar, Vegaswikian, Vipinhari, Visiting1, Vivio Testarossa, W Nowicki, Walton One, West London Dweller, Why Not A Duck, Winterspan, Wisamzaqoot, Wizardman, Wj32, Wutsje, Xezbeth, Ylem, Yogi raj2, Youssefsan, Zanimum, Zundark, Тиверополник, 522 anonymous edits

Bootstrap Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=461414029 *Contributors*: AlistairMcMillan, Andre Engels, Angela, Auntof6, B0at, Cburnett, Doanmanhtung.sc, Dr Zak, Feureau, Frap, Gamera2, Gogo Dodo, Greenrd, Hansa11, Idcmp, Ilario, Ilgiz, Imperfection, JTN, Jcw69, Jkl, Karl-Henner, Kate, Kbolino, Kbrose, Kvng, Little Professor, M4gnum0n, Magnus.de, Matt Crypto, MeltBanana, Mintleaf, Najoj, Netsnipe, Nuno Tavares, Orbst, Pion, Plinkit, R. S. Shaw, RedWolf, Rob Hooft, Ross Fraser, SWAdair, Schmid, Sietse Snel, Sir Norris, Skarebo, Stephan Leeds, Stiiixy, TakuyaMurata, The Anome, The Inedible Bulk, The Mysterious X, Valermos, Wbenton, Wikisierracharlie, Wrs1864, Zoicon5, 61 anonymous edits

Checksum *Source*: http://en.wikipedia.org/w/index.php?oldid=463616421 *Contributors*: Abdull, Agarlock, Alansohn, Aldie, Alex43223, Arvindn, Avihu, BNSDRX09, Badly Bradley, Bentogoa, Billybobbilly4325, Boothy443, CD-Host, CanisRufus, Chexum, Ciphergoth, Conversion script, Cuddlyable3, Damian Yerrick, DanielPharos, Davemckenna01, Dicklyon, Diddi, Drugonot, Dysprosia, DéRahier, Eloy, Entropicorder, EskilA, FitzColinGerald, Frap, Frappucino, Fredrik, Freewol, Furrykef, Gdassieu, Gdr, GentlemanGhost, Giftlite, Gmarie42, Gregb, Ham Pastrami, Iamthedeus, JeffreyBKing, Jhodapp1, Joel Saks, Jonelo, Jorge Stolfi, KLLvr283, Karnesky, Ksempac, Lacrimosus, Linas, Lowellian, MER-C, Matt Crypto, Melchoir, MementoVivere, Mh, Mindstalk, Mintleaf, Mitch Ames, Mormegil, Nakon, Navicular, Nealmcb, Neelix, Newarrior, Newell311, Oleg Alexandrov, Ozden, Peterjam70, Rick Burns, Rjwilmsi, Robofish, Salgueiro, Samw, Sedimin, Sg227, ShaunMacPherson, Simon Shek, Spiritllama, Stalfur, Sterrys, Takanoha, Taw, The Anome, Tobias Hoevekamp, TomViza, Traxs7, Vonvon, Vrenator, Wadlooper, Wayiran, Wolfch, Ykhwong, Zoobiewa, 151 anonymous edits

Circuit switching *Source*: http://en.wikipedia.org/w/index.php?oldid=458845897 *Contributors*: A.Petrenko, A.wasylewski, Acather96, Alex.muller, Andre Engels, B4hand, Basil.bourque, Bellhead, Ben4mai, Bevo, Biasoli, BradBeattie, Cburnett, Chetvorno, Danutz, Dgtsyb, Dthomsen8, Długosz, Eastlaw, EdH, Ee02b022, Episteme-jp, Eric, Fleminra, Foelectric, Frap, Gobbleswoggler, Harryzilber, Ilario, JTN, Jim.henderson, JohnOwens, JohnTechnologist, JonHarder, Jpatokal, Kbrose, Kvng, Lawrence Cohen, Lihui912, Lilac Soul, Lmatt, Looxix, MK8, MacStep, Mange01, Markrod, Mojodaddy, Mxn, NormanEinstein, Olof nord, Onceler, Orange Suede Sofa, Pearle, Piet Delport, Rfl, Rocketgoat, Rodran, Secretlondon, Shadowjams, Slowking Man, Sricciar, StaticGull, Tagishsimon, Thparkth, Thunderboltz, Tompsci, Turnstep, Vajira102, كاشف عقيل, 104 anonymous edits

Connection-oriented communication *Source*: http://en.wikipedia.org/w/index.php?oldid=442470434 *Contributors*: Abdull, Ataub2qf, AvicAWB, DuncanHill, Jflabourdette, Jim.henderson, JonHarder, Jradix, Kawawa.chitiya, Kbrose, Kvng, MacStep, Malomeat, Mange01, Marcuscalabresus, Nrubdarb, Quintus.bal, Radagast83, Sietse Snel, Spl, Sricciar, Storkk, W33v1l, Yrjö Kari-Koskinen, 21 anonymous edits

Connectionless communication *Source*: http://en.wikipedia.org/w/index.php?oldid=441834832 *Contributors*: AFool, Abdull, Eastlaw, Frencheigh, Gary D, Giftlite, Guy Harris, Jerryobject, JonHarder, Kbrose, Kevin Rector, Kgfleischmann, Kvng, Marcuscalabresus, MartinHarper, Michael Hardy, R'n'B, RJHall, Radagast83, Radiojon, Rita.dos.santos, Sietse Snel, Tedernst, VdHamer, Viriditas, Woohookitty, 15 anonymous edits

Cyclic redundancy check *Source*: http://en.wikipedia.org/w/index.php?oldid=463253082 *Contributors*: -aw-, AManWithNoPlan, Abdull, Aeris-chan, Afabbro, Algebra, AlphaPyro, Analogkidr, Applefat, Arghman, Arkrishna, Arny, AxelBoldt, Azfuzzball, Bartosz, BenRG, Bender235, Benhoyt, Bernard François, Bevo, Bhadani, Blah42, Bpburns88, Brian Geppert, Bruce1ee, Bsadowski1, C45207, CALR, Camitz, CanisRufus, Canyouhearmenow, Capricorn42, Carbonrodney, Charles Matthews, Chrisobyrne, Chuunen Baka, Ciphergoth, Cometstyles, Crl620, Csiebert, DMJ001, Dabadab, DagErlingSmørgrav, Damian Yerrick, Damieng, Danger, Davidgothberg, Davis685, Dcoetzee, Ddxc, Dgtsyb, Dicklyon, Dlohcierekim, DonDiego, Dubel, E235, Ed Brey, Eddie Parker, Efalk, Ekftsa, Elektron, Elkman, Emvee, Enigmaman, EoGuy, Erik Eckhardt, Eyreland, FML, Farpov, Fennec, Feulus, Fightin' Phillie, Flowersofnight, Frappucino, Fresheneesz, Fritzpoll, Fromageestciel, Fshahriar, Furrykef, Gary King, Gauss, Gerbrant, Ggia, Ghettoblaster, Giftlite, Gioto, Glenn, Gmaxwell, Grick, Groovy12, Gwernol, H1voltage, Haipa Doragon, Ham Pastrami, Herbythyme, Hyad, Iain Cheyne, Ihope127, Inkling, Intgr, Itusg15q4user, Ivan Akira, JackSchmidt, JakeVortex, Jakuzem, Jauerback, Jay314, JayPurohit, Jayrde, Jdoty, Jesse Viviano, Jgharston, Jhbdel, Jhfrontz, JimKi, Jjlin, Jmorgan, Joel Saks, Jokes Free4Me, Jonelo, Josh Patterson, K5002, KWParker, Karada, Karnesky, Kcho, Keilana, Kelapstick, Kelly Martin, Kgfleischmann, Klippa, KnightRider, Ktotheipi, Kurt.smolderen, L Kensington, L337p4wn, Lexus39, Linguica, LizardJr8, Lordfuzzz, MC10, MER-C, Magnus.de, Malke 2010, ManN, Marc Mongenet, Mark Renier, MarkMLl, Martin Rizzo, Martin451, Matt Crypto, MattTrivisonno, Maxal, Mayur, McSly, Mcsaenz, MementoVivere, Michael Hardy, Mifter, Mihir3445, Mintleaf,

Misheu, Motif user, MrOllie, Msa11usec, Multithreading, Nageh, Naohiro19 revertvandal, NawlinWiki, Nbarth, NewBitmaster, Nickdesbarats, Nikitadanilov, Noxia, Noxious2020, Nturton, Nuno Tavares, Nybbler, Oakad, Olathe, Oli Filth, Olivier, Ospalh, Osquar F, OutputLogic, Oxnard28, Paulschou, Pedro, PenitentSerf, PeterPearson, Piast93, Plasticspork, Plouton, Plvml6, Pne, Poindexter Propellerhead, Power piglet, Praveentech, Psb777, Psychotic Spoon, Qliedtke, Qwerty0, R'n'B, RDBrown, Rabbler, Rainbow will, Ralf.Baechle, Razimantv, Rchandra, Regregex, Rekinser, Rich Farmbrough, RichardTector, Rick Burns, Rumtscho, SAE1962, SLMarcus, SLi, Sam Hocevar, SciCompTeacher, Sfisher, Shadowjams, ShaunMacPherson, Shevek, Simaocampos, Simonhalsey, Skarg, Smyth, Spitzak, Sterrys, Stuffyvoid, Superm401, Surturz, Svansa, Svdb, TangentCube, Teles, Thadius856, The Anome, Timonczesq, Tobias Bergemann, Tomaxer, Trevor MacInnis, Tyler Oderkirk, Typo47, Tyrol5, Utcursch, Utezduyar, VMS Mosaic, Van helsing, Vanished user 34958, Veganfanatic, Voomoo, Vrenator, W2bh, Werbrouck, West London Dweller, Whosyourjudas, Wikibofh, William Pietri, Wk muriithi, Wmahan, Wrs1864, Wtshymanski, Yamamoto Ichiro, Yurik, Zhengchen, Zsinj, Zundark, ²¹², 569 anonymous edits

Data link layer *Source*: http://en.wikipedia.org/w/index.php?oldid=465384259 *Contributors*: 1000Faces, 3DS Mike, AAA!, AGruntsJaggon, Aldie, AlexandriNo, Alfio, Amillar, Angela, Arastcp, B4hand, Barri, BigDunc, Blehfu, Butko, Cahoover, Canterbury Tail, Casey Abell, Cgelpi3, Crispmuncher, DanMS, DavidABraun, Dddddd2w32, Dgtsyb, Dicklyon, Diogenes00, Docu, Dominio, Eekoo, Egil, Ehn, Ekabhishek, Ember of Light, EnOreg, Enjoi4586, Excirial, Falkonry, Gasheadsteve, Geek2003, Giftlite, Guy Harris, Hadal, Hede2000, Ibarrere, Intgr, Invictus42, Itusg15q4user, J.delanoy, Jafeluv, James smith2, Jmkim dot com, Johnuniq, Kannadigahavi, Kbrose, Ken l lee, Kjetil r, Konstable, Kubanczyk, Lawrence Cohen, Looxix, Lsukari, Luís Felipe Braga, Mange01, Martijn Hoekstra, MartinHarper, Mathieumcguire, Michael Hardy, Mlpearc, Nixdorf, Nolispanmo, Numa, Oli Filth, Omargamil, Philip Trueman, Plasticup, Playstationman, Rabarberski, Ravensun, RedWolf, Remember the dot, Rick Sidwell, SatyrTN, ScottDavis, Shanel, Shiro jdn, Sin-man, SirPavlova ♥, SkipHuffman, SpaceFlight89, Sross (Public Policy), Stdazi, Suruena, Talinus, Template namespace initialisation script, TexasAndroid, TheFeds, Thierryc, Tide rolls, Timo Honkasalo, Toccata quarta, Tomchiukc, TripleF, Turian, TutterMouse, Udunuwara, WLU, Webgeek, Weregerbil, Why Not A Duck, Widefox, Willy on Wheels over Ethernet, Wplacek, Yacht, Yyy, ZeWrestler, Zfr, Тивероподник, 173 anonymous edits

Error detection and correction *Source*: http://en.wikipedia.org/w/index.php?oldid=461232231 *Contributors*: 0nlyth3truth, 4th-otaku, Abdull, Aboeing, Agentbla, Anon lynx, Antaeus Feldspar, Apantomimehorse, Arthena, Arvindn, Auntof6, Avocade, Awmorp, Babybahamut, Beamishboy, Bekant, Bigs slb, Binary TSO, Bird of paradox, BitJam, Bob.v.R, Bobblewik, Bryan Derksen, Bumm13, Caesura, Cantalamessa, Cerireid, Chekholko, Cihan, Conversion script, Copeland.James.H, Cryptonaut, Cunchem, D6, DGerman, DanielPharos, DavidCBryant, DavidCary, Davidhorman, Deflective, Dogposter, Drhex, Dthede, Dysprosia, Eastlaw, Em3ryguy, Eric119, EugeneZelenko, EverGreg, Eyreland, Firien, Flying hazard, Francs2000, Gaius Cornelius, GearCutter, Giftlite, Gmazeroff, Gnorthup, GringoCroco, Gwern, Helder.wiki, Icecats11, Inventio, Iridescent, Izab1, Jaidan, Jao, Jcea, Jdseymour, Jerry, Jkl, Joey7643, Jorge Stolfi, Kenyon, Linas, Lir, Lotje, Lowellian, Mandarax, Mange01, Margin1522, Martarius, Masterpiece2000, Materialscientist, Mathuranathan, Michael Hardy, Mitchan, Miterdale, Mmeijeri, Morten, Music Sorter, MuthuKutty, Mysid, Nageh, Netpilot43556, Niceguyedc, Nihiltres, Nubiatech, Ojl, Oleg Alexandrov, Oli Filth, Phantomsteve, Phr, Pinethicket, Piotrus, Pnm, Poeloq, Pol098, Pouya, PrologFan, Pudhed, R'n'B, Radagast83, Ranjithsutari, Rbrewer42, Reedy, Reetep, Requestion, RexNL, Rgray001, Ricky81682, Ring0, Rjstott, Ruud Koot, Ryan Roos, SDS, SNIyer12, Sepia tone, Seven of Nine, Sikilai, Sixequalszero, Snowolf, Snoyes, Splash, Splintercellguy, Spoon!, Suffusion of Yellow, Supercoop, Swerdnaneb, T-rex, Taral, Technopilgrim, Teejayo, Tejas81, The Anome, The Rationalist, The Thing That Should Not Be, Tobias Bergemann, Traal, Trachten, Ubunaki, Unco, Vipinhari, WatchAndObserve, WestwoodMatt, Woohookitty, Ykhwong, ZeroOne, 228 anonymous edits

Ethernet *Source*: http://en.wikipedia.org/w/index.php?oldid=465173877 *Contributors*: 0612, 121a0012, 1lovegal, 1stJahman, 4a6f656c, 7, ARC Gritt, Aapo Laitinen, Abdull, Adashiel, AdjustShift, Adrian.benko, Aeluwas, Ahoerstemeier, Ajraddatz, Akriasas, Alan Liefting, Alansohn, Aldie, Ale jrb, Alecv, Algocu, AlistairMcMillan, Altermike, Alvestrand, Alyssapvr, Amatulic, Ameliorate!, Amillar, Amore proprio, Anaraug, Andareed, Andreas Toth, Andybryant, Andyhodapp, Angela, Angusmca, Anss123, Applemacintosh10, Arch dude, Arkrishna, Armando, ArnoldReinhold, Artaxiad, AtomEdge, Avono, Azaghal of Belegost, B4hand, BD2412, Bbachrac, Bctwriter, Bdmcmahon, Becritical, Betacommand, Betbest1, BiT, Bidabadi, Bigboymcgee, Bigjimr, Bilbo1507, Biot, BlueAg09, Blutrot, Bobblewik, Boffy b, Bomac, Bookbrad, Bookofjude, Bootleggingly mcbootleg, Boscobiscotti, Bovineone, Branclem, Branko, Brian Patrie, Btilm, Bumm13, Buy P%E%P%S%I, C0nanPayne, CRGreathouse, CS46, Calltech, Caltas, CamTarn, Can't sleep, clown will eat me, Capricorn42, Carlo.Ierna, Casey Abell, Catgut, Causantin, Cburnett, CesarB, Chaosgate, Charles Gaudette, Chasingsol, Chealer, Chmod007, Chowbok, Cjdaniel, ClementSeveillac, Closedmouth, CloudNine, Cluth, Cmdrjameson, Colin Marquardt, Conversion script, Corpx, Corti, Corwin8, Cpl Syx, Creidieki, Crispmuncher, Crissov, CryptoDerk, Cstanners, Cybercobra, DVdm, Daa89563, Dachshund, Damian Yerrick, Davehard, David Biddulph, DavidBailey, DavidH, Dcorzine, Deflective, Dennis Brown, Dicklyon, Digitalsushi, Discospinster, Diskdoc, DocWatson42, Dogcow, Donnieleefarrow, Download, Drak2, Drphilharmonic, Dtcdthingy, EEgirl18, EagleOne, Ebear422, Econrad, Eeekster, Efa, Egil, Ehn, Eivind F Øyangen, El Cubano, ElBenevolente, Electron9, Elektrik Shoos, Eleuther, Elinruby, Eloil, Engineerism, Enjoi4586, EoGuy, Epbr123, Evil genius, Ewlyahoocom, Excirial, Facts707, Femto, Fenrisulfr, Fetofs, Fieldday-sunday, Finest1, Firsfron, Flewis, FocalPoint, Frankchn, Frap, Fratrep, Fred Bradstadt, Fredrik, Fresheneesz, Fudoreaper, Fulldecent, Furrykef, G4wsz, GTAKIllerEric, Gah4, Gascreed, Geek2003, Gekkoblaster, Generica, Ghaly, Giftlite, Glenn, Gnalk, Gnuish, Gogo Dodo, Golgofrinchian, Gopher23, Goplat, Gosub, Graham87, Grandsonofmaaden, Greylion, Grgan, Guanxi, Guy Harris, Gyanlakhwani, H0serdude, HCelik, Haakon, Hadal, HappyCamper, Harryzilber, Harvester, Helix84, Herberthuber, Heron, Hobophobe, Hughey, Hungrymouse, Hussein 95, Hvn0413, Iambk, Iamzemasterraf, Ibc111, Idkyididthis, Igoldste, Indefatigable, Intgr, Iridescent, Irq, Isaac Rabinovitch, Itai, Itusg15q4user, J.delanoy, J04n, Jakohn, JamesBWatson, JamesEG, JeLuF, Jec, Jemichel, Jhartmann, Jim.henderson, John a s, JohnOwens, JohnWittle, Johnblade, Johnnyboyshoots, Johnuniq, JonHarder, Joseph507357, Joyous!, Jtkiefer, Juansempere, Judzillah, Juliancolton, K45671, KGasso, Kakomu, KaragouniS, Karn, Kate.woodcroft, Kbh3rd, Kbolino, Kbrose, Keith D, KelleyCook, Kerotan, Kevmcs, Kgfleischmann, Khalad, Kinema, King of Hearts, Kjkolb, Knucmo2, Kuru, Kvng, Kwamikagami, Kwiki, Lamro, Laudaka, Lavenderbunny, Lee Carre, Leebo, Leotohill, Lightmouse, Ligulem, LilHelpa, Limbo socrates, LittleBenW, LittleOldMe, Lockg, LordJumper, Lotu, Lovecz, Lped999, Lukith, Lumos3, Luna Santin, MER-C, MacGyverMagic, Mahjongg, Makeemlighter, Manish soni, Markjx, Martarius, Martijn Hoekstra, Matt.farina, Maury Markowitz, Mav, Mbutts, Melvinvon, Mendel, MessiFCB, Mhare, Michael Hardy, Michael Slone, Michael Thomas Sullivan, MightyWarrior, Mightyms, Mike Rosoft, Mikm, Milan Keršláger, Mindmatrix, Minesweeper, Mmiszka, Modulatum, MoussePad, Moxfyre, Mozzerati, MrFish, MrOllie, Mrand, Mrh30, Mwanner, Myanw, N TRoPY, NJM, NawlinWiki, NeaNita, Nelson50, NewEnglandYankee, Nicolaasuni, Nightraider0, Nimiew, Nishkid64, Nixdorf, Norm, Northamerica1000, Now3d, Nroets, Nthep, Nubiatech, Nvj, Nyttend, Oakad, Ohconfucius, Ohnoitsjamie, Oneocean, OverlordQ, OwenX, Oxymoron83, Packetslinger, Palmer1973, Paul, Paul Koning, Peck123, Pekaje, Percy Snoodle, Peruvianllama, Petrb, Peyre, Pgallert, Pgan002, Phil Boswell, Philip Trueman, Piano non troppo, PierreAbbat, Pilatus, PinkMonkeys, Pjbrockmann, Pkirlin, Plasticup, Plugwash, Pluma, Pmsyyz, Praetor alpha, Prolog, Publicly Visible, QmunkE, Quarl, Quintote, RHaworth, RadioFan, Rait, Rapaporta, Rarmy, Recurring dreams, RedWolf, Rednectar.chris, Reedy, Requestion, Rettetast, Rhobite, Rholton, Rich Farmbrough, RichardBennett, Rick Sidwell, Ricsi, Rjwilmsi, Rlcantwell, RoySmith, Roybadami, Royote, Rror, Rsrikanth05, Rufus210, Ruud Koot, Rw4nd4, Rwwww, SF007, SHCarter, Salsa Shark, Sam Hocevar, Sander123, Sceptre, Schneelocke, Scott McNay, ScottJ, SeaChanger, Selmo, Setu, Sfisher, Shadowjams, Shanes, Sharkford, Shieldforyoureyes, Sietse Snel, Silenceisfoo, Sincoskie, Sligocki, SoSaysChappy, Sobelbob, SpamBilly, Spiritia, Steeev, Stephan Leeds, Stephenb, Steven Luo, Stratocracy, Stuart P. Bentley, Suruena, Svick, Swapdisk, Swarnabhra, TAnthony, Tas50, Tedernst, Template namespace initialisation script, Tempodivalse, Thaas00, The Anome, The Random Editor, The Thing That Should Not Be, The quark, The undertow, TheMoog, Thingg, Thue, TimBovee, Tiny plastic Grey Knight, Tizio, TomPhil, Tonsofpcs, Tooki, Tothwolf, Tpbradbury, Transmission 1000, Trev M, Tunnie, Turgan, Tverbeek, Typo47, Tyz, UU, Ukexpat, Ulric1313, Unschool, Urvabara, Useight, Utuado, Vashti, Vdm, Veinor, Versus22, Vidmes, Vijaykumar, Viridae, Vmenkov, W Nowicki, Wa2ise, WadeSimMiser, Wasisnt, Wavelength, Wayfarer, WaysToEscape, Wbenton, Wefa, WhiteDragon, Wik, WikiJaZon, Wikiborg, Wikifranz, WillAndrews, WillLord, Willy on Wheels over Ethernet, Wimt, Wingnutamj, Wk muriithi, Wmahan, Wmasterj, Woohookitty, WriterHound, Wrs1864, Wtshymanski, Xenium, Xnatedawgx, Yintan, Yudiweb, Yuvalnod, Yyy, ZenerV, Zero10one, Zhangyue, Zodon, Zoicon5, 975 anonymous edits

Ethernet frame *Source*: http://en.wikipedia.org/w/index.php?oldid=464417702 *Contributors*: Alexh19740110, Bostonvaulter, Electron9, EoGuy, Guy Harris, Hawaiian717, Intgr, Jiemingwei, Kvng, Mange01, Mayur, QueBurro, Quinn d, Shashiskills, Tchai, W Nowicki, Wjwatson67, Xezbeth, Yemisilin, 49 anonymous edits

Ethernet hub *Source*: http://en.wikipedia.org/w/index.php?oldid=464768884 *Contributors*: (boxed), A930913, ActivExpression, Alai, Alan Liefting, Alan Rockefeller, Alansohn, Alaudo, Ali@gwc.org.uk, Alvestrand, AxelBoldt, B4hand, Beetstra, Biot, Bloodrain, Bobblewik, Borgx, Can't sleep, clown will eat me, Canterbury Tail, Carlhazlewood, Cholmes75, Chutzpan, CliffC, Coasterlover1994, Codman20, Coren, Cweath7, Dcirovic, Demicx, Dfrt5, Djcater, Donfbreed, Epbr123, Esanchez7587, Fraberj, Frencheigh, Frungi, Gah4, Gdo01, Gerald.combs, Giftlite, GraemeL, Guy Harris, Havarhen, Hcberkowitz, I dream of horses, Icairns, Indefatigable, J.delanoy, Jaan513, Jade Knight, Jamielappin, Jdthood, Jef-Infojef, Jim.henderson, Johnuniq, JonHarder, Jusdafax, Kvng, Leuko, Lightmouse, LilHelpa, Little Mountain 5, LlamaLlamaLamp, Llamadog903, Luna Santin, Maestrosync, Mange01, Matthew V Ball, Maximaximax, Mayur, Mehra, Michael B. Trausch, Mmmeg, NewEnglandYankee, Nimakha, Nnila, Northernhenge, Oli Filth, Orborde, Oxymoron83, P996911turbo, Pak21, Pamri, Petr Kopač, Pevalou, Pilatus, Pinkadelica, PizzaofDoom, Plugwash, Pnm, Puat133, Quaeler, Quarl, Queen Rhana, Rick Sidwell, RobertG, RyanCross, S3000, Samuelsen, Seddon, Senator Palpatine, Shotgunlee, SlimVirgin, Smadge1, Some Old Man, SquidSK, Stdazi, Stee20000, Suresh p, Swanny92, Tide rolls, Trevor Spiteri, ULTRA001, UU, UncleBubba, Unschool, Uršul, Vanished 6551232, Waggers, Walkerguy, Weregerbil, Wikievil666, Wknight94, XSTRIKEx6864, Zac67, 311 anonymous edits

Frame (networking) *Source*: http://en.wikipedia.org/w/index.php?oldid=436864929 *Contributors*: 16@r, Aleksey Gerasimov, Barcex, BigFatBuddha, Bookandcoffee, Brewcrewer, Casablanca2000in, David Haslam, Debresser, Deville, Dysprosia, Eyreland, Ghaly, Hadal, Ixfd64, JonHarder, KnowledgeOfSelf, Kvng, LachlanA, Luna Santin, Mac, Mange01, Missionary, Nancy, Nuno Tavares, Patrick, Phatom87, Rocketgoat, Sephiroth storm, Statkit1, The Anome, TripleF, Twinkie Assassin, Waskage, Willy on Wheels over Ethernet, Wk muriithi, Wmasterj, Zack, Σ, 38 anonymous edits

Frame Relay *Source*: http://en.wikipedia.org/w/index.php?oldid=452666449 *Contributors*: 0x6adb015, A p3rson, A-hiro, Academic Challenger, Acprisip, Afita, Ageo020, Agou, Aldie, Andareed, Apyule, Ardafif, B4hand, Bellhead, Bjankuloski06en, Blind cyclist, Bobblewik, Borgx, Brholden, Brianhe, Bsdguru, Cacophony, Caltas, CattleGirl, Cburnett, Chcknwnm, Cwacker1, Dgtsyb, Diannaa, Djg2006, Dogposter, Dragon Dave, Editore99, Elykkyle, Engineerism, Eric Scace, Gaius Cornelius, Gascreed, Gildemax, GraemeL, Guy Harris, HarisM, Hellbus, Ichimonji10, Igorchubin, IlyaHaykinson, Intgr, Itpastorn, JDX, JForget, Jj137, JohnAlbertRigali, Johnteslade, JonHarder, Karl-Henner, Kbrose, Kocio, Kthnxrick, Kvng, Leonardopsantos, Lightmouse, Lilac Soul, Lumos3, Mam711, Mange01, MarkWahl, Mdsam2, Metricopolus, Michael Hardy, Michaelas10, Mild Bill Hiccup, Minesweeper, Mitack, Mmmeg, Mrugeshthakkar, NicoNet, Nivix, Noctibus, Obel, Onceler, Pedant17, Phatom87, Philip Trueman, Ppelleti, Prunesqualer, Pwnage97, Quarl, Ruwanindika, Sam Hocevar, Sgarson, Shahid789, SimonEast, Sindala, Sintagma, SnappingTurtle, Snowolf, Studerby, T1Rex, The Anome, ThreeBlindMice, Tolien, UberScienceNerd, VandalCruncher, Vargenau, Vicenarian, Visiting1, Vlospf, Wj32, Xrgtn, Zahnrad, 248 anonymous edits

Frame synchronization *Source*: http://en.wikipedia.org/w/index.php?oldid=443595695 *Contributors*: Colin Marquardt, Eastlaw, Jim.henderson, JordoCo, Msa11usec, Nabla, Noted trip3, RclintonH, The Anome, Ulyssix, 10 anonymous edits

Gateway (telecommunications) *Source*: http://en.wikipedia.org/w/index.php?oldid=451950438 *Contributors*: *drew, Ajo Mama, Al Lemos, Alinja, Alphos, AnnaP, Antandrus, Arastcp, ArglebargleIV, Biasoli, Boscobiscotti, Brewthatistrue, Cetinsert, Damian Yerrick, Demicx, Deville, Dillysj, EmersonLowry, ExNihilo, Fourohfour, Funandtrvl, Furkhaocean, Gtrmp, Harryzilber, Helmoony, Ian Pitchford, Iranway, J.delanoy, Jacobko, Johannes100, Jojhutton, Junckerg, Kateshortforbob, Koolbear, Kved, Mandarax, Marokwitz, Matthew Fennell, Mulad, Nbarth, Nscheffey, Nuno Tavares, OlEnglish, Pascalv, Pbb, Pearlgearl, Pedant17, Pikiwyn, R'n'B, Radagast83, Slogsweep, Tabletop, Tbhotch, Thomcraver, Unyoyega, W Nowicki, Woohookitty, X42bn6, Zhou Yu, 106 anonymous edits

Internet protocol suite *Source*: http://en.wikipedia.org/w/index.php?oldid=464338629 *Contributors*: 130.243.79.xxx, 203.109.250.xxx, 213.253.39.xxx, 66.169.238.xxx, A8UDI, Aapo Laitinen, Abdull, Abdullais4u, Acceptus, Acroterion, Aeonx, Ahoerstemeier, Alansohn, Albanaco, Aldie, Ale2006, Alek Baka, AliMaghrebi, Aliasptr, Alireza.usa, AlistairMcMillan, Amungale, Ana Couto, Anna Lincoln, Anororn, Arcenciel, ArchonMagnus, Arteitle, ArticCynda, Avant Guard, Avicennasis, Axcess, AxelBoldt, B4hand, Barberio, Barnacle157, Bender235, Bentogoa, Bernard François, Betterworld, Bezenek, Bhavin, Biot, Bloodshedder, Bmicomp, Branko, Brian.fsm, Brion VIBBER, Camw, Canthusus, CaptainVindaloo, Carnildo, Casey Abell, Cate, Cburnett, Chadernook, Cheesycow5, Ckatz, Coasting, Conversion script, Coolcaesar, CrinklyCrunk, Ctm314, Cybercobra, Cynthia Rhoads, DARTH SIDIOUS 2, Damian Yerrick, Daniel Staal, DanielCD, Darkhalfactf, DavidDW, DavidDouthitt, Denisarona, DerekLaw, Dgtsyb, Dicklyon, Disavian, Dmeranda, Dnas, Dogcow, Doradus, Dorgan65, Doug Bell, Drphilharmonic, Duffman, EagleOne, Ed g2s, Edmilne, Edward, Edwardando, Eeekster, Ekashp, Electron9, Ellywa, Elwood j blues, EncMstr, Enjoi4586, Epbr123, Eptin, Equendil, Ericl234, Erik Sandberg, Ethanthej, Etu, Evil Monkey, Evil saltine, Expensivehat, Falcon9x5, Ferkelparade, Fixman88, Fr34k, Freyr, GaelicWizard, Geneb1955, Gilliam, Glane23, Glenn, GlobalEdge 2010, Globemasterthree, Golbez, GordonMcKinney, Graham87, Gringo.ch, Gsl, Guy Harris, Haakon, Hadal, Hairy Dude, HarisM, Harryzilber, Hcberkowitz, Headbomb, Helix84, Here, Hoary, Holylampposts, Hpnguyen83, Hyad, IMSoP, Ilario, Imcdnzl, Imran, Indeterminate, Inhumandecency, Inomyabcs, Intgr, Itai, J.delanoy, JTN, Jackqu7, James Mohr, JamesBWatson, Jantangring, Jatkins, JesterXXV, Jimp, Jmdavid1789, Jnc, Joanjoc, John Vandenberg, Johnblade, Johnuniq, JonHarder, Jorunn, Jrogern, Jsoon eu, Jusdafax, JustAGal, KYPark, Kaare, Kasperd, Katieh5584, Kbrose, Kim Bruning, Kim Rubin, KnowledgeOfSelf, Konman72, Koyaanis Qatsi, Krauss, Krellis, Kungming2, Kusma, Kvng, Kyng, Labongo, Larree, Law, Layer, Leapfrog314, Lee Carre, Logictheo, Lova Falk, Luna Santin, Magister Mathematicae, Maltest, Mandarax, Mange01, Manop, Marcika, Martyman, Martyvis, Master Conjurer, Matt Dunn, Mattbrundage, Matthew Woodcraft, Matusz, Mav, Mckoss, Mechanical digger, Mendel, Merlissimo, Metaclassing, Michael Hardy, Miles, MilesMi, Mintguy, Mothmolevna, Mrzaius, Mukkakukaku, Mwarren us, Mzje, NMChico24, Nasz, Navedahmed123, NawlinWiki, Nealcardwell, Nealmcb, NewEnglandYankee, Ngriffeth, Nhorton, Nick C, Niteowlneils, Nivix, Nixdorf, Nknight, Nmacu, Northamerica1000, Nubiatech, Nv8200p, Obradovic Goran, Oheckmann, Olathe, Otets, OttoTheFish, Oxwil, Oxymoron83, Palfrey, Papadopa, Patilravi1985, Paul, Paul Koning, Paulkramer, Perfectpasta, Peripitus, Pfalstad, Pharaoh of the Wizards, Phgao, Piano non troppo, PioM, Plugwash, Pokeywiz, Poslfit, Pps, Public Menace, Punjabi101, Putdust, Quinxorin, R'n'B, RaNo, Radagast83, Ramnath R Iyer, Rayward, Reaper Eternal, RedWolf, Reliablesources, RevRagnarok, Rich Farmbrough, Rich257, Richardwhiuk, Rick Block, Rick Sidwell, Rjd0060, Rjwilmsi, RobEby, RobertG, RobertL30, Roberta F., Robost, Rodeosmurf, Ross Fraser, Rrelf, Rserpool, Runis57, Ruthherrin, SJP, STHayden, SWAdair, Samjoopin, SasiSasi, Seaphoto, Sheldrake, Shii, Shiraun, Shiro jdn, Shizhao, Sietse Snel, Sjakkalle, Skullketon, Smartse, Smjg, Snaxe920, SnoFox, Spmion, Stahla92, Stefan Milosevski, Stephan Leeds, Stephenb, Suffusion of Yellow, Sully, SuperWiki, Suruena, Svick, Swhitehead, Swpb, Ta bu shi da yu, Tagishsimon, Tarquin, Techmonk, Techpro30, Template namespace initialisation script, Tfl, That Guy, From That Show!, Thatguyflint, The Anome, The Nut, TheOtherJesse, Theresa knott, Thingg, Thumperward, Thunderboltz, Thw1309, Tide rolls, Tim Watson, Timwi, TinaSDCE, Tmaufer, Tmchk, Tobias Hoevekamp, TomPhil, Topbanana, Tr606, Tyler, Typhoon, Ukexpat, Unyoyega, Vanis314, Vegaswikian, Victor Liu, Violetriga, W163, Wadamja, Waggers, Weregeek, West.andrew.g, Weylinp, Whereizben, Wiki104, Wikid77, Wikiklrsc, William Avery, Wimt, Wolfkeeper, Wonderstruck, Woohookitty, Wrs1864, XJamRastafire, Xeesh, Xojo, Xosé, Yakudza, Yas, Ydalal, Yudiweb, ZNott, Zac439, Zeerak88, Zfr, Zigger, Zoicon5, Zondor, Zundark, Zvezda1111, ^demon, شات صوتي, 787 anonymous edits

Label switching *Source*: http://en.wikipedia.org/w/index.php?oldid=396890788 *Contributors*: Bearcat, Betacommand, Greenrd, Loadmaster, Malcolma, MarkMLl, PigFlu Oink, Punk9706, Storkk, Sunil rane, Valley2390, Y2kcrazyjoker4, 1 anonymous edits

LAN switching *Source*: http://en.wikipedia.org/w/index.php?oldid=459274220 *Contributors*: AlexandriNo, Ali farjami, Angela, ArnoldReinhold, Boscobiscotti, Calliopejen, Dgtsyb, Drag09, Edward, Elfguy, Fang Aili, Infrangible, Jeremy Visser, Jim.henderson, Kappa, Kjkolb, Lee, M2petite, Mani1, Materialscientist, Mxn, Nikola Smolenski, Nneuman, Perry Bebbington, Piper8, Pradneshpatil, Rcsloan75, Rich257, Riick, Robost, Sachinjain2011, Shahid789, Shenme, Tellyaddict, Thumperward, Tom Herbert, Underpants, Vaibhavbamne1, Vitamin77, Wadamja, Woohookitty, 36 anonymous edits

Link layer *Source*: http://en.wikipedia.org/w/index.php?oldid=460984076 *Contributors*: CapitalR, Chzz, David Woodward, Dicklyon, Enjoi4586, Fred Gandt, Isomorphic, Jacobko, Jafeluv, Jirka6, Jujutacular, Kbrose, Mange01, McGeddon, MrInfo2012, Psychlohexane, R'n'B, Rafostry, Sophus Bie, Suruena, Wrs1864, 13 anonymous edits

Logical link control *Source*: http://en.wikipedia.org/w/index.php?oldid=462314763 *Contributors*: Anrie Nord, BaomoVW, Bearcat, Beno1000, Borgx, Chester Markel, Dgtsyb, Dicklyon, Discospinster, Doulos Christos, ErikHaugen, Frap, Fsiler, GermanX, Guy Harris, Hgfernan, Indil, Itusg15q4user, Kauczuk, Kbrose, KelleyCook, Krmboya, Laurusnobilis, Mange01, Materialscientist, Muhandes, Od Mishehu, Pgallert, Phatom87, Ppchailley, Qoqnous, The Anome, Very Input, Wayfarer, Widefox, 53 anonymous edits

Media access control *Source*: http://en.wikipedia.org/w/index.php?oldid=464499744 *Contributors*: @modi, A purple wikiuser, Alexander.stohr, Angela, Arathald, Arkrishna, Avitesh, Beno1000, Bentogoa, Borgx, Breno, Caesura, Casey Abell, Cburnett, Closedmouth, Crazy Murdoc, Dawnseeker2000, DenesVadasz, Dicklyon, Djg2006, Dominio, Ebraminio, Enjoi4586, ErikHaugen, Evercat, Extraordinary, Figureskatingfan, GPHemsley, Ghez, Giftlite, Gwalker nz, HappyCamper, HarisM, Intgr, Irnavash, Itai, JFSM, Jesse Viviano, Joconnor, Johnuniq, Jusdafax, Kbrose, KelleyCook, KnightRider, Leon Hunt, Lihui912, Loosecannon93, Luís Felipe Braga, Mange01, Mellery, Mlaffs, Mojodaddy, Mr Stephen, Nageh, Notheruser, Nuno Tavares, Nurg, OrangeDog, Pgallert, Philip Trueman, ProveIt, Rick Sidwell, Snori, Stassats, Steven Hepting, Suruena, Template namespace initialisation script, Timwi, Treekids, Vitz-RS, Widefox, Willy on Wheels over Ethernet, Woohookitty, Youssefsan, Zachlipton, Zanetu, Zoicon5, 103 anonymous edits

Multiplexing *Source*: http://en.wikipedia.org/w/index.php?oldid=464200426 *Contributors*: 16@r, 28421u2232nfenfcenc, ARC Gritt, AbsolutDan, Ahmad87, Alan Canon, Aldie, Atlant, B4hand, Blue520, Bluemouse2306, Bobo192, Booyabazooka, Bravelymistaken, Brofski, Carbon-16, Catslash, CyberSkull, Dancter, Daniel C, Daniel.Cardenas, DavidCary, Deville, Drbreznjev, Ecxelectronics, Editor at Large, Ekoontz, Endlessnameless, Epbr123, Equant, FireyFly, FreeKresge, Frogtumor, Ggia, Gpvos, GraemeL, HEL, Harryzilber, Hydrargyrum, Hypercube, J. M., JamesBWatson, Jclemens, Jer71, Jim.henderson, Jj137, Jonathan Drain, JordoCo, JorgeGG, Josh Parris, Karada, Karakal, Kbrose, Kc307, Kjkolb, LOL, Leandrod, Lee Carre, LeoNomis, LilHelpa, M.e, MMuzammils, Magnus.de, Mandarax, Mange01, Marie Poise, Markaci, Mathiscool, Mdsam2, Michael Hardy, Mikerockwell1975, Mjb, Mlewis000, Mo ainm, Myanw, Mysdaao, Nbarth, Nikola Smolenski, Nsaa, PanagosTheOther, Pearle, Pen1234567, Poweroid, Radiojon, Rdsmith4, ResidueOfDesign, Reslfj, Rich Farmbrough, RobertG, Roman12345, Ronz, Sadi Carnot, Sarath02, Satbuff, Searchme, Seraphimblade, Serinde, SpiderJon, Steven J. Anderson, StubbyT, Tawker, Tedickey, The Thing That Should Not Be, TheAMmollusc, Tompsci, Tpobrien, Uncle G, Uzume, Wbm1058, Yazid966, Youssefsan, Zachlipton, Тиверополник, 184 anonymous edits

Network switch *Source*: http://en.wikipedia.org/w/index.php?oldid=465273360 *Contributors*: A8UDI, Abune, Abyss of enchantment, Acroterion, Adalie, Adam850, Adamantios, Aditsu, Aegicen, Aero Leviathan, Airman99, Akc9000, Alansohn, Aldie, Aldis90, Ale jrb, Alexandria, Alpha 4615, Alvestrand, Alykhan2006, Andre Engels, Androsyn, Angus Lepper, Apparition11, Arpabone, Arthena, Astronautics, Atlantia, Atomician, Atropos235, Auntof6, B4hand, Barry26, Bencherlite, Berkut, Bjankuloski06en, Blashyrk, Bluemoose, Bluerasberry, Blues-harp, Boscobiscotti, Bovineone, Breno, Brfiner, Brnsuga99, Brons, Brucevdk, Btyner, Butseriouslyfolks, C45207, Cagatay.ucar, Capnquackenbush, Caruch6392, Casey Abell, Cassini83, Chiefwei, Chmod007, Chowbok, ChrisCork, Clemente, Colinstu, CommonsDelinker, DARTH SIDIOUS 2, DVD R W, Danielcohn, Defendfreedom, Demicx, Dgtsyb, Djg2006, Dkhydema, EJvL, Eitheladar, Electron9, Equendil, Erich gasboy, Evil saltine, Excirial, Favonian, Fleung, Flowanda, Fmcway, Forseti, Fred J, Freeingmind, Frosty3219, Fuzheado, Fyyer, Fæ, Gamera2, Geimas5, Giftlite, Glenn, Guy Harris, H2g2bob, Hcberkowitz, Helix84, Heymid, Hunppa, II MusLiM HyBRiD II, Iambk, Ibc111, Illythr, Indefatigable, Intgr, Itusg15q4user, J.delanoy, J.smith, JHunterJ, JPINFV, JamesHoadley, Jasper Deng, Javawizard, Jbridge19, Jdthood, Jef-Infojef, Jim.henderson, John Vandenberg, Johnuniq, JonHarder, Jsharpminor, Julesd, Julie188, Kadin2048, Kaschei, Kayvee, Ke4roh, Kesvas, KnowledgeOfSelf, Kostmo, Koyaanis Qatsi, Kozuch, Kvng, Laveol, Lenilucho, Li4kata, Lightmouse, Marco255, Marek69, Mateo2, Matthodge77, Melaen, Memset, Mentifisto, Misteror, Mitar, Mmmeg, Mo ainm, Moozaad, Motley Crue Rocks, MrOllie, Mxn, Napalm Llama, Nathanael Bar-Aur L., Natkeeran, Neilc, Nelson50, Nestea Zen, NickBush24, NickSentowski, Nicolaasuni, Nimakha, Ninly, NonNobisSolum, Nubiatech, Nuno Tavares, OllieFury, Omegatron, Omicronpersei8, Opelio, P4k, Paddy lib, Pamri, Panchurret, Patcat88, Paul 012, Paul Koning, Paul Slocum, PaulHanson, Pawl Kennedy, Peripitus, Perl, Peter Horn, Petiatil, Pgilberd, Pharaoh of the Wizards, Phatom87, Philip Trueman, Philjk4eva, PierreAbbat, Piesangboer, Pipedreambomb, Piper8, PlayCold, Plugwash, Pnm, Pxma, Quarl, R'n'B, Rachmaninoff, Raucanum, Rchandra, Reaper Eternal, Reconsider the static, RedDragonAkai, RedWolf, Requestion, Rich Farmbrough, Rick Sidwell, Rje, Roguelazer, Roystonea, Rror, Rsduhamel, Rtcpenguin, Rubicon, Sander123, Sannse, Seitz, Sephiroth storm, Shadowjams, Shahid789, Sharon gngt, Sheetalthebest, Sjmrox4ever, Sreeharim, Srinivasbt, Stevedouglas, Sunborn, Tbsdy lives, Tfga, Thaas00, The Anome, The Thing That Should Not Be, Theresa knott, Tide rolls, Tim Starling, Tim1988, Tizio, Tompsci, Torahulonly, UU, Ummit, UncleBubba, Uyer, Vegardw, Vi2, Visor, Voidxor, Voldemortuet, Vssun, W Nowicki, WWStone, WarrenA, Wikidudeman, Wikipelli, Will Hawes, WillNess, Winchelsea, Woohookitty, Wtmitchell, YUL89YYZ, Zac67, ZachCrichfield, Zhen Lin, 557 anonymous edits

Packet switching *Source*: http://en.wikipedia.org/w/index.php?oldid=464709637 *Contributors*: A.Petrenko, A.wasylewski, Airplaneman, Ali Esfandiari, AlphaPyro, Alxeedo, Andrewpmk, Barberio, Benhoyt, BertK, Bevo, Binksternet, Bp2010.hprastiawan, BradBeattie, Brewhahaitsme, Can't sleep, clown will eat me, Capricorn42, CarlHewitt, Cburnett, Cgarciap86, Cgdallen, Chancemill, Charles Matthews, Chris the speller, Ciphers, Closedmouth, Dave Cohoe, Dead3y3, DeadEyeArrow, Deema AlShamaa, Dgtsyb, Diberri, DonDaMon, Donreed, Edcolins, Ee02b022, Ewlyahoocom, Excirial, Fishyghost, Foelectric, Foobaz, Freedomfighter21, Fresheneesz, Fumblebruschi, GPHemsley, Gentleman wiki, Gerfaut, Giftlite, Groundeyes, Harryzilber, Hcberkowitz, Helix84, Hjf, Ilario, Itai, J.delanoy, JNW, JTN, Jamescfield, Jedi Master Brownlow, Jedonnelley, Jim.henderson, Jm34harvey, Jnc, JoanneB, JohnGrantNineTiles, Johnbojaen, JonHarder, Jopsen, Jpatokal, Jpbowen, Jy, K12u, Kabdcn, Kablammo, Karol Langner, Kbrose, Kungfuadam, Lee Carre, LeviathinXII, Logicat, LouScheffer, MCBastos, MacStep, Mange01, Manop, Maurreen, Mbarbier, Mendaliv, Merope, Mild Bill Hiccup, Mojodaddy, Neurolysis, Niteowlneils, Nmacu, Noctibus, Nubiatech, Nuno Tavares, Oli Filth, PerryTachett, Phatom87, Philbarker, Philip Trueman, PierreAbbat, R2jitu, Rabarberski, Rdmoore6, RedWolf, Reliablesources, Rich Farmbrough, Rspanton, Sowsnek, Sricciar, SunCreator, Synchrodyne, Teemu Maki, That Guy, From That

Show!, The Anome, The Original Wildbear, The Singing Badger, The Thing That Should Not Be, Thparkth, ThreePD, Timsk, Timwi, Tom94022, Toreau, Torla42, Usangel1066, Vary, Voyagerfan5761, Vrenator, W Nowicki, W163, Wa3frp, Waveguy, Welsh, William Avery, Wingsandsword, Wizard191, Wmasterj, Woohookitty, Yamaguchi先生, Ynhockey, Zondor, Тивероплник, 230 anonymous edits

Point-to-point protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464915640 *Contributors*: 66.57.5.xxx, Aether8m, Ahoerstemeier, Alansohn, Aldie, Am ran32, Amir, Andareed, Andre.goddard, Anon lynx, Anwar saadat, Bakanov, Becky Sayles, BertK, Bkkbrad, Bomazi, Borgx, Brighterorange, CLW, Calvin 1998, CanisRufus, Cgbraschi, Choosetocount, Christian Christelis, Comet Tuttle, Conversion script, CyberSkull, Cybercobra, DDerby, DavidBailey, Deysahmad, Dictabeard, Dmeranda, DonaldEastlake3, Dorcots, Dreish, EagleOne, Edaelon, Enigmaman, Ewlyahoocom, Fragglet, FrancoGG, Fred Gandt, Freeaqingme, Fumblebruschi, GK tramrunner, Gadfium, Ger-Jan, Gerald Squelart, Guy Harris, Gxojo, Hvtuananh, Itpastorn, JTN, Jesse Viviano, JoanneB, Jordav, Josh Parris, Juansempere, KJRehberg, Kinema, Kwamikagami, LarryGilbert, LiDaobing, Ligulem, Linkminer, Lmatt, Mac, Mancini, Mange01, MathsPoetry, Matusz, Mhare, MilkMiruku, Mipadi, Moretz, Moribunt, Muhandes, Negrulio, Nuno Tavares, Olivier, Omniplex, Oneiros, Pantergraph, PenguiN42, Peyre, Pgk, Phatom87, Plugwash, Pmsyyz, Pradzee123, PseudoSudo, Psz, Qbeep, R'n'B, Radagast, Random contributor, Rdj, Rfl, Rjwilmsi, Roadrunner, Roberta F., RossenV, RupertMillard, SMcCandlish, Sander Säde, Scott Paeth, Sdfisher, Searchme, Sega381, Shanes, Smilesfozwood, Sparky132, SquarePeg, StaticGull, Stirling Newberry, Stryke3, Superborsuk, Superskippy, Suppie, Synchrite, Tagishsimon, Tapir Terrific, Tcncv, The Anome, The Thing That Should Not Be, Thug666, Tietew, Timc, Tyrel, Uoyylno, User27091, Valenciano, Vikramrn, Vlospf, W Nowicki, WanderingHermit, Wayfarer, WikHead, William Allen Simpson, Yaronf, Yayay, Zondor, Тивероплник, 248 anonymous edits

Promiscuous mode *Source*: http://en.wikipedia.org/w/index.php?oldid=447757136 *Contributors*: AManWithNoPlan, Aenar, Amelia Hunt, B4hand, Betterworld, Bezzm, Blotwell, Brookshawn, Chris 73, Christan80, Cohesion, Dcoetzee, Diman011, Douglas Saraiva, Drago9034, Dreish, ElBenevolente, Elvey, Evil saltine, Excirial, Felipe1982, Flarn2006, Flockmeal, Frap, Funtaff, GPHemsley, Gnitset, Gunit31, Guy Harris, Gz33, Jamelan, Jcsutton, Jdm64, Jeremy Visser, John Vandenberg, Jon-emery, JonHarder, Joy, Lightdarkness, Mlaffs, Pgan002, Poindexter Propellerhead, Qhalilipa, R'n'B, Shadowjams, Shellreef, StenSoft, SunCreator, T.J. Crowder, Tbhotch, Tburket, The Anome, Tieno, UU, Woohookitty, Yuanli.H, 94 anonymous edits

Protocol data unit *Source*: http://en.wikipedia.org/w/index.php?oldid=458223646 *Contributors*: 4twenty42o, Aldie, Biasoli, Boscobiscotti, Crossmr, Devourer09, DylanW, Eastlaw, Excirial, Fromageestciel, Gaius Cornelius, GrahamDavies, Guy Harris, Hetar, Itusg15q4user, Jnc, Josh Parris, Joshurtree, Jpbowen, Knuckles, Kvng, L Kensington, Lexikorn, Liotier, Logictheo, Mancini, Mange01, Meredyth, Mitch Ames, Palmer1973, PeterEasthope, Piet Delport, R'n'B, Rhodekyll, Selfev, Tatrgel, ZeroOne, 43 anonymous edits

Proxy ARP *Source*: http://en.wikipedia.org/w/index.php?oldid=451810108 *Contributors*: Aldie, Bacteria, Bovineone, Btilm, Daves, Drewshin99, Ettrig, Gaius Cornelius, Giraffedata, Gurch, Iansk, Ilario, Johnuniq, Kamal444, Kd4ttc, LilHelpa, Lilprincesskrishna, MER-C, Manfreeed, Max dg, MessiFCB, Milan Keršláger, Mintguy, Moocha, Piano non troppo, Plugwash, Rick Sidwell, Sam Hocevar, The Anome, Upholder, Vmart, Woohookitty, 46 anonymous edits

Quality of service *Source*: http://en.wikipedia.org/w/index.php?oldid=465360532 *Contributors*: A.b.s, ASchmoo, Aarghdvaark, Acprisip, Adamantios, Adclark88, Agi896, Agou, Amatriain, Andkore, AnnaFrance, Azevedo wikipede, BCSWowbagger, Barek, Beck162, Benjaminzsj, Bissinger, Bondjb, Borgx, Bovineone, Broughturner, Btilm, Bunnyhop11, Camilo Sanchez, Chetvorno, Chris Capoccia, Christopher Mahan, ClockworkSoul, Closedmouth, Cmsjustin, Conversion script, Ctxppc, D.alotaibi, Dancter, Davesnmp, Dbfirs, Dgtsyb, Dinomite, Djg2006, Dmatsunaga, DocWatson42, Donalmorris, Dori, DéRahier, ENeville, Echoray, EdH, Elimerl, Elizant, Eljazzar, Epbr123, Equendil, Eraserhead1, Erencexor, Everyking, Foglar, Francesco-lelli, Frap, Fred Bradstadt, Glenn, Graham87, Grebaldar, H, HPNAUSER, Huismancljo, Ian Geoffrey Kennedy, Icey, Idragan, Iliev, Ireneo.heraldo, Itusg15q4user, Ixim dschaefer, JLeland, Jclemens, Jim.henderson, Jishnua, JoeSmack, John of Reading, Jordav, Karthikndr, Kbrose, Kegill, Kelcher, Kgfleischmann, Kinema, Kuru, Kvng, Kyle Barbour, Kyokpae, Lahnfeear, Lee Carre, Limeyuk, Lirneasia, Luci Sandor, MJisnotmylover, Mamyles, Mange01, Mannafredo, Manop, Mark Bergsma, Mark Tranchant, MartinHarper, Martnym, Martyvis, Mc6809e, Megaman en m, Michael Hardy, MichaelStanford, Mike Schwartz, Mindmatrix, MkDoyle, Morton.lin, MosesB77, Motor, Mozzerati, Muhandes, Nsk92, Ntavares, Nádvorník, Ohnoitsjamie, Oskar Liljeblad, PPBlais, Pbhurley, Phatom87, PhilipR, Pluggy35, Pol098, Pythoulon, RHaworth, Raanoo, Radagast83, Raindeer, RedWolf, Rememberway, Rewstah, Rich Farmbrough, Rich257, Richard n, RichardBennett, Rjwilmsi, Rkinder, Roberts83, Robertvan1, Salexandre, SasiSasi, Sceasary, Scott.somohano, Sd58619, Sfisher, Shotgunlee, Simetrical, Skc7, Skittleys, SkyWalker, Soulkeeper, Spainhower, Spakin, Spc01, Stevevogelnu, SunCreator, Suruena, Tclavel, The Anome, Tony Sidaway, Trusilver, Ttwaring, Uetian118, V2Blast, Vandenbr, W Nowicki, Wapcaplet, WhiteBoy, Whophd, Wik, Wk muriithi, Wolfkeeper, Xxp, Yamaguchi先生, Yaronf, YaseminITÜ, Zhou Yu, Zodon, 317 anonymous edits

Reverse Address Resolution Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=459575381 *Contributors*: Abisys, Aldie, BRUTE, Beno1000, Besho84, Bongdentoiac, Celinama, Druiloor, Dunerat, EagleOne, Ekeb, El C, Epolk, Feureau, Ffaarr, Fingew, Gamera2, Gilliam, Ilario, IlyaHaykinson, JTN, Jaapvstr, Johnuniq, Jpfagerback, Kbrose, Mange01, Mintguy, Mintleaf, Mkb218, Mosca, Motor, Nichtich, Niteowlneils, Philip Trueman, Rafostry, Rkinder, Rookkey, Rorro, Signalhead, Sohit14all, Wereon, 55 anonymous edits

Router (computing) *Source*: http://en.wikipedia.org/w/index.php?oldid=464935590 *Contributors*: 128.107.253.xxx, 16@r, 1wolfblake, 4.35.185.xxx, 4twenty42o, A8UDI, AKMask, AdamWeeden, Adamantios, Adamn, Aditya, AdjustShift, Adoniscik, Aecis, Aeusoes1, Afed, Ahoerstemeier, Ahy1, Ajo Mama, Akc9000, Alan Liefting, Aldie, Alex Smotrov, AlexFerrara, Alexandropulos, Algebra, Algocu, Alimaq9, Alpinwolf, Altermike, Alvin-cs, Alyssapvr, Amire80, Andy120290, Angr, Anonymi, Antandrus, Anthonyjameswood, Apparition11, Arch dude, Arnon Chaffin, Arpabone, Atchius, Avoided, BMB, Barneca, Bbukh, Bcat, Bebenko, Ben D., BenAveling, Benemin, Bernard S. Jansen, Betbest1, Bevo, BigChicken, Bishonen, Bjankuloski06en, Blood reaper, BlueH2O, Bobo192, Boing! said Zebedee, BorgHunter, Boscobiscotti, Brandon, Brendan Moody, Brianboonstra, Brianga, Bryan Derksen, BrydoF1989, Bunnyhop11, CWii, CZmarlin, Cactus.man, Camw, Can't sleep, clown will eat me, Canterbury Tail, Capricorn42, CardinalDan, CaribDigita, Carlwikicarl, Catbar, Causa sui, Cbacho, Cburnett, CecilWard, Chetvorno, Chexum, Chilipalmerlll, Chlodwig, Chris the speller, ChrisHodgesUK, Chun-hian, Chuq, Clemente, Cobi, Colonna, Cometstyles, CommonsDelinker, Conversion script, Cpl Syx, Crakkpot, CroweIO, DARTH SIDIOUS 2, DVdm, Damian Yerrick, Daniel.Cardenas, Darth Panda, David0811, DavidCiani, DeadEyeArrow, Deagle AP, Deepak365, Delirium, Demicx, Denniss, DerHexer, Desuz, Dgies, Dinko13, Discospinster, Dmt018, DocWatson42, Dpm64, Drhex, Dtrebbien, Eclipsettaja, Edokter, El C, Emote, EmporerD, English Fox, Enviroboy, Epbr123, Eric Shalov, Ethereal Dragon, Excirial, Extols, Extransit, F15x28, Fabricationary, Face, Facts707, Falcon8765, Favonian, Ferkelparade, Finereach, Firebee, Flewis, Francis E Williams, Frap, Furrykef, Fuzheado, Fvw, Fyyer, GB fan, GSSAGE7, Galoubet, Gareth Jones, Gascreed, Gazwald, Gcirafisi, Geek2003, Geimas5, Gidonb, Giftlite, Gjp23, Glenn, Goat-see, Gogo Dodo, Goodnightmush, Gracenotes, Graeme Bartlett, Graham87, Green451, Gtg204y, Gwernol, Gökhan, Hadal, Hairmare, Ham Pastrami, Hardwaredotcom, HarlandQPitt, Hashir tasleem, Hcberkowitz, Hdt83, Heavy1974, Heramba11, Hfszeto, Hibana, Hm2k, Honta, Hsmith254, Humannetwork, Hut 8.5, IRP, IRT.BMT.IND, ISJohns, Ian Lynch, Ianmacm, Ilario, Imran, Insanity Incarnate, Ireneshusband, Iridescent, Ironholds, Isatre, Itai, J Di, J.delanoy, JForget, JPilborough, JTN, Jacheem.m.eqbal, Jackrabbit2, James086, Jared Preston, Jasonfward, Jaxl, JdeJ, Jec, Jhdaly, JidGom, Jim.henderson, Jinlye, Jjron, Jnc, Jodi.a.schneider, Joe102283, JoeOnSunset, JohnT, JonHarder, Jrash, K Watson1984, K4DSP, Kakero, Kaszeta, Kbolino, Kernel.package, Kgfleischmann, Kgrr, Kingghost, Kizor, Kku, Klemen Kocjancic, Komap, Kremit, Kukini, Kuru, Kvng, Kwamikagami, L00pILaUzXxX, LachlanA, Leandropls, LeaveSleaves, Leebo, Leodmacleod, Lepidus16, Lerdthenerd, Liftarn, Lightmouse, LightningMultiCom, LilHelpa, Lilac Soul, Little Mountain 5, LittleBenW, Livrocaneca, Lloyd Wood, LonelyBeacon, Lotje, LouScheffer, Love manjeet kumar singh, Lowzeewee, Luna Santin, Lupo, Luxomni, MER-C, Mac, Mackensen, Magioladitis, Marek69, Markharris, Masonprof, Mateo SA, Materialscientist, Mattgirling, Matthew Martin11, Matusz, Mav, Mboverload, Meegs, Memset, Mgoerner, Miaow Miaow, Micahhiggs, Michael Devore, Michael Hardy, Midoladido, Mike1024, Millis, Minna Sora no Shita, Mintleaf, Mipadi, Mmernex, Mo0, Mohitsport, Monkeyman, Moonie0079, Moratinnz, Moulding, Mrx374325732, Mschlindwein, Muhandes, Mushroom, Mxn, Mygerardromance, Nakon, NaranKPatel, Nasnema, Natkeeran, Nbarth, NerdyScienceDude, Nethgirb, Netkinetic, Neuro, Neverquick, Ngriffeth, Nick, Niteowlneils, Njh, Nmacu, Nn123645, No Guru, NonNobisSolum, Nposs, Nubiatech, Nww mag, Octahedron80, Odie5533, Oli Filth, Omega187, Omicronpersei8, OrgasGirl, PJTraill, Palsmartboy, Papadopa, Paul Koning, Pbb, Pchov, Pdcook, Pedant17, Penosa, Petrasmom, Phil Holmes, Philip Trueman, PhilipMW, Phynicen, Piano non troppo, Pianodude363, Pinethicket, Polly, Possum, Potter831, Pradeep pn, Pradeepdesign, Prolixium, Protonk, Qrsdogg, Quinxorin, R srivastav2000, R'n'B, RTC, Raanoo, Raul654, Readro, RebelzGang, Rebroad, Retodon8, RexNL, Rhanbury, Rick Sidwell, Ricky@36, Rjwilmsi, Robertbricker, Robertpeters9, Roberts83, Robertvan1, Robthebob, Rohitthakral, Rollier, RonSigPi, Ronz, Rrjanbiah, S-n-ushakov, SJP, Sakhalinrf, Scootey, Seaphoto, Secretlondon, Seraphimblade, Shacharg, Shadowjams, Shahid789, Shanedidona, Shanes, Sharon gngt, Shervinafshar, Shortynugget, Shyamal, SilentC, Simetrical, SimonMackay, SirGrant, Skäpperöd, Slp1, Smalljim, Smappy, Sonicforest, Sorintopex, SpaceFlight89, Specs112, SpuriousQ, Sridev, Staka, Stephenb, Sternagel, Steveboy90, SummonerMarc, SunSw0rd, Tabasco01, Team4Technologies, TechNotch, Tedickey, Tekinfo, Telecomwisdom, The Anome, The Thing That Should Not Be, TheDwoo, Thej2, Thornomad, Thorpe, Tide rolls, Timberlax, Tinss, Tinucherian, Titoxd, Tobycat, Tollund man, Tomas skare, Tomgreeny, Tommcc7, Tommy2010, Tony1, Topdeck, Travisyoung, Triwbe, Trusilver, Tumbledry, TwistOfCain, Tylerdmace, Typhoonchaser, Tyrenius, UU, Uncle Dick, Unfree, Vanka5, Vassili Nikolaev, Vegaswikian, Verbatim9, Vhiskas, Via strass, Vijaykcm, Vishalsaran, Vishnava, Vjardin, Voicerouter, Voidxor, Vorthax, Vvarkey, W Nowicki, WLU, Waggers, Waxiemarie, Wernher, Western Pines, Wideangle, WikiFew, WikiLeon, WillMak050389, Willie the Walrein, WinTakeAll, Winkydink, Wlgrin, Wtmitchell, Wtshymanski, X42bn6, Xmp, Xtreme.msen, Xyzzy288, YUL89YYZ, Yamamoto Ichiro, Yath, Yelyos, Yendor1958, Yennathan, ZacBowling, Zachb90e, Zeroshell, Zpb52, Zwirello, 1173 anonymous edits

Self-synchronizing code *Source*: http://en.wikipedia.org/w/index.php?oldid=422050819 *Contributors*: Classicalecon, Eastlaw, Excirial, Jaan Vajakas, Jim.henderson, Magioladitis, Mange01, Nbarth, RadioFan2 (usurped), Sflubry, Viriditas, Zanimum, 12 anonymous edits

Shared medium *Source*: http://en.wikipedia.org/w/index.php?oldid=430196555 *Contributors*: A876, Fleebo, Fresheneesz, Jengelh, Jim.henderson, Keilana, XLerate, 2 anonymous edits

Spanning Tree Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=465290397 *Contributors*: 7, ABF, Aaron.hebert, AaronSw, Abrech, Abune, AdrianSuter, Aldie, Altenmann, Alvestrand, Alvin-cs, Ant1357, Anthony Appleyard, Apps123, Arnaud, Atigala2003, B4hand, Baccala@freesoft.org, BartonM, Bellenion, Bellhead, Bergonz, Blatkinson, Bongwarrior, Booyabazooka, Boscobiscotti, Bryan Derksen, Bstpierre, CALR, CableCat, Capricorn42, Charfles, Davepape, David Eppstein, Dicklyon, Dirk gently, Dominus, Dysprosia, DéRahier, EAderhold, ESkog, Easwarno1, Echoray, Edurant, Ellery, Emurphy42, Epbr123, Eug, Everyking, Evyn, Fabriciodosanjossilva, FlyingToaster, ForthOK, Frank, FrankTobia, Friday, Galoubet, Gardar Rurak, Ghettoblaster, Ghewgill, GhosT, Giraffedata, Glass Sword, Grafen, Guy Harris, Habbie, Harshilvyas, HorsePunchKid, Ilario, Irmatov, Ixfd64, J.delanoy, Jeffq, Jeroenr, Jkl, JonHarder, Joseanes, Kai-Hendrik, Kaushal.shakya, Kb, KelleyCook, Kgrr, Krellis, Kvng, Larry Yuma, Lee Carre, Lhmathies, Lightmouse, LimoWreck, Lotje, MER-C, MadSurgeon, Mchaddock, Mcicogni, Mephistophelian, Michael Hardy, Michael Slone, Miken32, Miketubby, Mincebert, Minna Sora no Shita, Mmeerman, Negrulio, Ngriffeth, Nickfox, Nuno Tavares, Os87, P00r, Palthainon, Pdpatil, Pgallert, Pgr94, Phisches, Plugwash, Pmaccabe, Pmj, Pnm, Populus, PsyberS, RJHall, Radagast83, Rafterman, Redfearnb, Rednectar.chris, Reinderien, Rjwilmsi, RockMFR, Salems19, Sanddune00, Sceptre, Scobra77, SeanDague, Seanx820, Shahid789, Skandha101, Skier Dude, Stagira, Stryqx, Suruena, Tfdb, The Thing That Should Not Be, The emm, Tigey, Timwi,

Tinucherian, Tkteun, Tofergregg, Triddle, UncleBubba, Vlhsrp, Weirdy, WikiReviewer.de, Wk muriithi, Yellowking, Yidisheryid, Zack, Zigger, 324 anonymous edits

Tunneling protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464475688 *Contributors*: 16@r, Accurizer, Alessandro57, Arthena, Bposert, Bruns, Cavan, Ccostis, CesarB, Chris D Heath, Cwolfsheep, Drummle, EagleOne, Ebelular, Edgar181, Emesee, Enjoi4586, ErikWarmelink, Exlex, Fleminra, Gforce20, GraemeL, Grimboy, Hcberkowitz, Hégésippe Cormier, Indil, IronJohnSr, JYOuyang, Jamelan, JamesAM, Jandalhandler, Jerome Charles Potts, JidGom, Joshuaterrill, Jtg, Jules.lt, Karnesky, Kbrose, Kizor, Kku, Klaus100, Knuckles, Kobekatt, Larytet, Laug, Liangent, Lusitana, Mail2yogie, Mange01, Materialscientist, Matt314, MementoVivere, Ministry of Truth, Mjb, Mozzerati, MrOllie, Nachoman-au, Negrulio, NickPenguin, Noahspurrier, PabloCastellano, Palfrey, Pazkooda, PeterZacSmith, Plat'Home, RJFJR, Reelrt, Rl, Rohitkalaskar, Sietse Snel, SkipHuffman, Sourabhforu, Starunj, Subsolar, Surabh7, Suruena, Sysy, Tclavel, The Anome, The MAZZTer, ThreeBlindMice, Watson1966, Winterst, 94 anonymous edits

Type of service *Source*: http://en.wikipedia.org/w/index.php?oldid=456063831 *Contributors*: 1ForTheMoney, Beeblebrox, Bestitem, Beterhans, Bomazi, Cpetty-wiki, Dandorid, Fram, Gimboid13, Gudeldar, Harryboyles, Hashar, Huwlewis, Jec, Juansempere, Kappa, Kelly Martin, Kvng, Monedula, Rich Farmbrough, Scientus, Tony1, TubularWorld, Weylin.piegorsch, WikiTome, Ynhockey, Zil, 43 anonymous edits

Virtual circuit *Source*: http://en.wikipedia.org/w/index.php?oldid=458267070 *Contributors*: 49oxen, BD2412, BenAveling, Bernard François, Björn König, CYD, DasBub, Dcirovic, Dgtsyb, Dto, EdH, Emiller213, Imran, Jamelan, Jh559, Jim.henderson, Mange01, Michael Hardy, Minna Sora no Shita, Nikola Smolenski, Niteowlneils, Phatom87, Punk9706, Radagast83, Radiojon, Rameshbabunt, Rivertorch, Santryl, Sbhujbal, Souravsachin, Tagishsimon, UncleDouggie, 54 anonymous edits

Autonomous system (Internet) *Source*: http://en.wikipedia.org/w/index.php?oldid=452452869 *Contributors*: 28421u2232nfenfcenc, Abune, Adzinok, Babajobu, Betacommand, Borgx, Bwoodcock, Charles Matthews, Cmdrjameson, Coppertwig, Cwolfsheep, Cybercobra, Dan100, Duckbill, EmreDuran, Erkan Yilmaz, Etu, Fl, Fudoreaper, Furtim, Gerbrant, Giftlite, Gobbleswoggler, HCBerkowitz, Hashar, HorsePunchKid, Hughesey, IW.HG, Infofarmer, Isoma, JMS Old Al, JTN, Jag123, Johnuniq, Josh Parris, Kbrose, Ke4djt, Kenyon, Kgfleischmann, Kingdon, Ksn, Mandarax, Michael Hardy, Mjuarez, Niteowlneils, Nneonneo, Omarecd, Oneiros, OverlordQ, Pap3rinik, Petri Krohn, Pgilman, Phantomsteve, Pharaoh of the Wizards, Pmsyyz, Qr.brown, RG2, Radagast83, Schneelocke, Scolobb, Serych, SiD3WiNDR, Sigkill, Silvarbullet1, Snaily, The Anome, Timpala, Tjwagner, Ttaylor249, Wrs1864, Петър Петров, 95 anonymous edits

Best effort delivery *Source*: http://en.wikipedia.org/w/index.php?oldid=464716487 *Contributors*: Burbble, Itai, Johnteslade, Karlhauth, Kauczuk, Kvng, MBlue2020, Mange01, Manicsleeper, Mannafredo, Martin Fasani, Nard the Bard, Phatom87, PoisonedQuill, QuantumEngineer, Scriberius, Tangentidea, TitaniumDreads, Tomek.poznan, Welsh, Xanzzibar, 11 anonymous edits

Broadcast address *Source*: http://en.wikipedia.org/w/index.php?oldid=465154730 *Contributors*: Baykalpolat, Betterworld, Bob.v.R, Capricorn42, DKEdwards, Dysprosia, Eddpayne, Fargue, Frap, Gnuish, Ilario, JTN, Jengelh, Jleedev, JonHarder, Kbrose, Kencaruso, Kosebamse, Krellis, Kvng, Lear's Fool, MER-C, Mschlindwein, Muzzl3, RedWolf, Shaddack, SixSix, SuperHamster, Tbsdy lives, Unbound, Vd437, Whateveryousaysir, Zac439, 51 anonymous edits

Broadcasting (computing) *Source*: http://en.wikipedia.org/w/index.php?oldid=452951257 *Contributors*: 2doorsdown, Aeon1006, Baccala@freesoft.org, Can't sleep, clown will eat me, Casey Abell, CrescentCaren, Easyas12c, Henk.muller, Ixfd64, JonHarder, JordoCo, Joris, Kbrose, Kvng, M4gnum0n, Oicumayberight, Paintman, Panleek, Phatmonkey, Storkk, SymlynX, Wihwang, 23 anonymous edits

Classful network *Source*: http://en.wikipedia.org/w/index.php?oldid=462116262 *Contributors*: Aaa3-other, Aaaa140383, Adoniscik, Alansohn, Aldie, AliBi, AlistairMcMillan, Allefant, Andonic, Andrewcrawford, Armando, Arthena, Atomiktoaster, AySz88, Azrael81, Betterworld, Bilogik, Bovineone, BruceLueckenhoff, Bwagstaff, Chzz, Corti, Courtarro, Crh0872, Crimson30, DariuszT, DarthStrydre, DerEikopf, DerHexer, Dr.queso, EJDyksen, El C, Elebis, Ernest lk lam, Ethanzzz, FJPB, Fewaffles, Fluffernutter, Gamera2, Gautam217, Gfoley4, Hairy Dude, Happysailor, Heron, IainP, Ik 5, Ilario, Indefatigable, Ivansanchez, J.delanoy, JTN, Jengelh, Jeremiahsavage, JeremyA, Jerroleth, Jerrymmanteca, Jgateau, Jmundo, Jnc, Johnuniq, Jonathan Drain, Jz32300, Kbdank71, Kbrose, Kjetil r, Krellis, Kshieh, Kvng, Leo vegoda, Looper5920, Loscri, LostTemplar, Lowellian, Mathieu Bonnet, Mikeblas, Mindmatrix, Mro, Mumiemonstret, NCurse, Nerd42, Nethac DIU, Nick2588, Ninly, Nubiatech, Ohnoitsjamie, P0lyglut, Panarchy, Passionless, Patar knight, Phidjit, PhilKnight, Platypus222, Prunesqualer, Psz, Qjim, RadioFan2 (usurped), Rbonvall, Rick Sidwell, Savabore, SineSwiper, Skakkle, Smyth, Suruena, Suso, Tbhotch, TerokNor, The Anome, ThePianoMan, Thoglette, Tizio, Tobias Bergemann, Tohd8BohaithuGh1, Tonywalton, Tr606, Twitsed, Txmcse, Vlhsrp, Wangi, Whisky drinker, Wikipelli, Woohookitty, Wrs1864, ZeroOne, Zondor, 355 anonymous edits

Classless Inter-Domain Routing *Source*: http://en.wikipedia.org/w/index.php?oldid=464920995 *Contributors*: .anaconda, 4wajzkd02, AGK, Abrahamvegh, Abune, Ajaykpr123, Alaniaris, Aldie, Alvestrand, Andreas S., Ashdurbat, Aurigas, Baccala@freesoft.org, Bigmantonyd, Borgx, Bradipus, Brianhe, Brolin Empey, CQJ, Callidior, Calliopejen1, Callmederek, Cbdorsett, Cburnett, Cdc, Celerityfm, Conversion script, Crossmr, Crotalus horridus, Csernazs, Cwolfsheep, DanielSiva, Dart evader, DataWraith, Dinomite, Dogcow, Doria, El C, Ellmist, Fabriciodosanjossilva, Falcon8765, Feezo, Frey, Fuenfundachtzig, Fuhghettaboutit, Googol plex, Graham87, GregorB, Hcberkowitz, Horselover Frost, Hu12, I K Brunel, IOLJeff, Infofarmer, Intgr, Isidore, Itai, JTN, JackTinWNY, Jamelan, Jantman, Jasper Deng, Jberkes, Jen2000, Jengelh, Jerry, Jnc, Joconnor, Joonga, Jp.martin-flatin, Jz32300, Kbdank71, Kbrose, Keckster675, Kefengx, Kenyon, Kgfleischmann, Klykken, KuRiZu, Kubigula, Kvng, Lbruno, Levyonthelevee, LittleDan, LogicX, Lucky13pjn, Maetrics, Mange01, Markiemarkos, Melnakeeb, MementoVivere, Michael Hardy, Mikewax, Mindmatrix, Miskaton, Mnbf9rca, NSR, Neilc, Niteowlneils, No1lakersfan, Nomenclator, OlivierM, OwenBlacker, Paul, Paxsimius, PedroPVZ, Phill.jolliffe, Phy1729, Quuxplusone, Radagast83, Radu - Eosif Mihailescu, Rait, Rich Farmbrough, Rmcgowan1548, Robert Brockway, Rycee, Samjac, Sdrtirs, Shawnhath, Sleightly82, SpectatorRah, Spinningspark, Stevertigo, Suruena, Suso, SymlynX, TangentCube, Tenretnieht, The Anome, The wub, This user has left wikipedia, Tobias Bergemann, Viridae, Wk muriithi, Wrs1864, X21J, YBG, Zapyon, 242 anonymous edits

End-to-end principle *Source*: http://en.wikipedia.org/w/index.php?oldid=455993142 *Contributors*: A7N8X, Abach, AdamRoach, Aldie, Amina.alobaid, Atomician, Brz7, Camilo Sanchez, Cybercobra, Dcljr, DerBorg, EugenZ, Fubar Obfusco, Gunnar Guðvarðarson, Hu12, JTN, Jabowery, Jhawkinson, JonHarder, Karada, Kbrose, Khalid hassani, Kvng, Lexspoon, Ma7moud.asp, Mbaer3000, Mezzaluna, Milhouse fr, Miym, Mlewis000, Neilc, Niteowlneils, Oita2001, Prowsej, Rememberway, RichardBennett, Roadnottaken, SasiSasi, Sengkang, Shoeofdeath, Sivar, SystemBuilder, The Anome, Thumperward, WibblyLeMoende, Wolfkeeper, Zvar, 60 anonymous edits

Holddown *Source*: http://en.wikipedia.org/w/index.php?oldid=441844921 *Contributors*: Celinama, Chrisa11, Ddimitrov, Gtrmp, JonHarder, Minusf, Piet Delport, RL0919, Rjwilmsi, Royce, Scope creep, Whitepaw, Woohookitty, 11 anonymous edits

Hop-by-hop transport *Source*: http://en.wikipedia.org/w/index.php?oldid=446802026 *Contributors*: Drilnoth, Jim.henderson, Morki, Phatom87, Shmlchr, 2 anonymous edits

Host (network) *Source*: http://en.wikipedia.org/w/index.php?oldid=453680942 *Contributors*: 1exec1, Akc9000, Astramaglia, Bento00, Biasoli, Chriscandy, Cuziyam, Epbr123, Ewlyahoocom, Galoubet, ImperfectlyInformed, J04n, Jnc, Johnuniq, Kbrose, Kyng, Mange01, Nethgirb, PPerviz, Pharaoh of the Wizards, R'n'B, RockMFR, Rsocol, Sesshomaru, Wayiran, 17 anonymous edits

ICMP Redirect Message *Source*: http://en.wikipedia.org/w/index.php?oldid=445503109 *Contributors*: AHands, AJWright, Beowulf king, Bradycardia, Jbohac, Kenyon, RedWolf, Saxbryn, Sietse Snel, Snjoetw, Woohookitty, Wrs1864, 8 anonymous edits

Internet Control Message Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=463434648 *Contributors*: Adamianash, Ajk91, Alerante, Alvin-cs, Andareed, Anggarda, ArielGold, Athaenara, B20180, Beowulf king, Bgraabek, BobHackett, Bradycardia, Caesura, Celtkin, Chealer, Conversion script, Courcelles, Dark knight ita, DeadEyeArrow, Dnevil, DrThompson, Drj, Electrocut, Enjoi4586, Face, Fbriere, Forton, Fresheneesz, Fubar Obfusco, Goreatic, Graham87, Grenavitar, Gwern, Hari36533, Hawaiian717, Huwr, J.delanoy, JTN, Jadhav.m, Jasminek, Jay-Sebastos, Jengelh, Jnc, Joseanes, Kbrose, Kgfleischmann, Kinema, LAX, Mahboud, Mange01, Matusz, Mayonaise15, Meand, Mintleaf, Monkeymaker, Monkeyman, Mr link, Mremail1964, Naveenpf, Nimiew, Niteowlneils, Nixdorf, Nuno Tavares, Omar35880, Oskilian, PeterB, Phatom87, Piper8, Plustgarten, Pmsyyz, Pointy haired fellow, Polluks, Poweroid, Promethean, Quibik, Qwertyus, RHaworth, Ranjiths, Rjgodoy, Romanm, SarahEmm, Shellreef, Sietse Snel, Sioux.cz, SixSix, Suruena, Template namespace initialisation script, The Anome, Tjpayne, Tobias Bergemann, Tstojano, TuukkaH, UncleBubba, Unixguy, Verdatum, VictorianMutant, Voidxor, Wayne Hardman, Whbstare, Wikisierracharlie, William Avery, Woohookitty, Wrs1864, Xmm0, YordanGeorgiev, 210 anonymous edits

Internet Group Management Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=460734021 *Contributors*: AS, Abolen, Accumulator one, AlephGamma, Antichrist, Apexprim8, B4hand, Bktero, Cgcastor, Charles Matthews, DWay, David Newton, DéRahier, Ed g2s, Enjoi4586, Evan, Falcon8765, Grasshoppa, Grunzh, Gwalla, Hairy Dude, Harpreet82, Hede2000, Hoo man, JTN, Kaal, Karada, Kbrose, Kenyon, Kgfleischmann, Kinema, Kvangend, Kvng, Leolaursen, Mange01, Mateo2, Metaclassing, Msiebuhr, Nichtich, Nikai, Nothing1212, Patrickdepinguin, Pchov, Pmolinero, Psu256, R. S. Shaw, R.srinivaas, Raanoo, Rcronk, RedWolf, Robhu, Runkalicious, Ruwanindika, Salgueiro, Sango123, Suruena, SvartMan, Technobadger, Thumperward, Trasz, Wikihelp1, Woohookitty, Zoicon5, Тивополник, 114 anonymous edits

Internet Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464756848 *Contributors*: 2w133, A-moll9, A. B., ARUNKUMAR P.R, Abb615, Abdull, Adagio Cantabile, Addihockey10, Aggelos.Biboudis, Ahoerstemeier, Aldie, Altesys, Alvin-cs, Andareed, Andre Engels, Andywandy, Angelo.biboudis, Antiuser, Anttin, Anwar saadat, Ardonik, Arkrishna, Attilios, BAxelrod, Badanedwa, Bdesham, Beefman, Bennor, Bentogoa, Biot, Bjornwireen, Blanchardb, Blehfu, Blue520, Bobguy7, Bobo192, Borislav, Brest, Brianga, Bridgecross, Brim, Brouhaha, Bryan Derksen, Bryanbuang1993, CALR, Calltech, Camilo Sanchez, Capricorn42, Caramdir, Carlo.Ierna, Casey Abell, Cbdorsett, Cburnett, Cdyson37, Ceo, CesarB, Chealer, Chenzw, Christian List, Closedmouth, Conversion script, Coolcaesar, Coralmizu, Corruptcopper, Courcelles, Cverska, Cwolfsheep, Cybercobra, Cyclonenim, Cynthia Rhoads, DARTH SIDIOUS 2, Dan D. Ric, Daniel Staal, Daniel.Cardenas, DanielCD, DeadEyeArrow, Defyant, Demian12358, Denisarona, Dmaftei, Dnas, Doria, DrBag, Drugonot, Dwheeler, EagleOne, Echuck215, Eclipsed, Eequor, El C, Elfguy, Enjoi4586, Epbr123, Erodium, EverGreg, Evil saltine, FGont, Felipe1982, Ferkelparade, Fish147, Florentino floro, Formulax, Forton, Fredrik, FrummerThanThou, Fæ, Galoubet,

Gamera2, GaryW, General Wesc, Giftlite, Glane23, Glenn, Goatbilly, Graciella, GraemeL, Grafen, Graham87, Hairy Dude, Hardyplants, Harvester, Hayabusa future, Helix84, Hemanshu, Hetar, Hughcharlesparker, I am Me true, Iamregin, Imcdnzl, Imroy, Intgr, Ixfd64, J.delanoy, JHolman, JTN, Jack Phoenix, Jadounrahul, JamesBWatson, Jasper Deng, Jauhienij, JavierMC, Jdforrester, Jeff Carr, Jheiv, Jiddisch, Jimgeorge, Jimys salonika, Jnc, Jno, JohnGrantNineTiles, JonHarder, Justsee, KD5TVI, Karl McClendon, Karol Langner, Katalaveno, Kbrose, Kgfleischmann, Kim Bruning, Kocio, Krellis, Kubigula, Kukini, Kvng, L Kensington, Latitudinarian, Liangent, Lion789, Looxix, Lotje, Love manjeet kumar singh, Maerk, Mahabub398, Mailtomeet, Mammad2002, Mange01, Manop, ManuelGR, Markr123, Marr75, Max Naylor, Melter, Mike Rosoft, Mikieminnow, Mindmatrix, Mion, NGNWiki, Naive cynic, Nakon, NawlinWiki, NeoNorm, Nialldawson, Nightraider0, Nimiew, Nixdorf, Noformation, Noldoaran, Northamerica1000, Nubcaike, Nubiatech, Nugzthepirate, O, Ogress, Olathe, Otolemur crassicaudatus, Oxymoron83, Paolopal, Patrick, Paul, PaulHanson, PedroPVZ, Phatom87, Philip Trueman, Pinkadelica, Piotrus, Plustgarten, Poeloq, Poweroid, Ppchailley, Python eggs, Rabbit67890, Recognizance, Reedy, Reub2000, Rgclegg, Rich Farmbrough, Richard Ye, Rjgodoy, Rkrikorian, RobertG, Robocoder, Rsduhamel, Ryamigo, Ryanmcdaniel, Sandstein, SasiSasi, Scientizzle, Scientus, Shiftoften66, Shlomiz, Shoeofdeath, Sir Arthur Williams, Sjaak, SkyLined, SmilingBoy, Smyth, Some jerk on the Internet, Sonicx059, Sophus Bie, SpaceRocket, SpeedyGonsales, Stephenb, Stevey7788, Supaplex, Suruena, Sysiphe, THEN WHO WAS PHONE?, TakuyaMurata, Tarquin, Teles, Template namespace initialisation script, Tezdog, The Anome, The Transhumanist (AWB), Thorpe, Timwi, Tiuks, TkGy, Tobias Bergemann, Tommy2010, Torla42, Trevor MacInnis, Tristantech, TutterMouse, UncleBubba, Urvabara, V-ball, Varnav, Vary, Versageek, Versus22, VillemVillemVillem, Voyagerfan5761, Warren, Wayiran, Wayne Slam, Wcourtney, Wereon, Where, Whitepaw, Wik, Wikibob, Wikisierracharlie, William Avery, Wknight94, Woohookitty, Wrs1864, Wtmitchell, Yahoolian, Yamamoto Ichiro, Yausman, Yidisheryid, Yyy, Zarcillo, Zundark, Zzuuzz, شات صوتي, පසිඳු කාවින්ද, 493 anonymous edits

IP address *Source*: http://en.wikipedia.org/w/index.php?oldid=464747448 *Contributors*: *drew, 09ialsharrai, 0waldo, 1.mallesh, 16@r, 1a2b3c4e5d, 203.109.250.xxx, 20percent, 21655, 48states, 4twenty42o, 5 albert square, 76df457hjkozdfg, 7qrfl41r, 9258fahsflkh917fas, A Softer Answer, ABF, AVand, Abdul muqeet, Abecedare, Abmitgkp2011, Acalamari, Acather96, Accurizer, Acroterion, Adammw, AdjustShift, Aervanath, AgainErick, Ageekgal, Agent007bond, Agentscott00, Agurzil, Ahkitj, Ahoerstemeier, Ahy1, Airconswitch, Airplaneman, Aitias, Ajuk, Alan Rockefeller, Alanbrowne, Alansohn, Alex Cohn, Alex43223, Alison, AlistairMcMillan, Alpha 4615, Alphachimp, Alphax, AltecLansing12, Alvestrand, Amnuay, Andres, Andrewlp1991, Andrewski, Andy, Andy Dingley, Andy Smith, AngelOfSadness, Anna Lincoln, AnonGuy, Anonymous editor, Antandrus, AnthonyR, Anthonymetal, Apoyon, Applesqsx, Aqberr, Arakunem, Archer7, Argentium, ArglebargleIV, Arizona1983, Arman Cagle, Arthuran, Artibaton, Ashenai, Asoiaf fanatic, Atlant, AtomSmazher, Avnjay, Avogadro94, Avoided, Avono, AzaToth, Baccala@freesoft.org, Badmachine, Barek, BarretBonden, Bayberrylane, Benwildeboer, Berro9, Beta m, Betbest1, Bfolkens, Biblbroks, Bigjimr, Bihco, Bill37212, BioPupil, Bjankuloski06en, BlackAce48, Blake3522, Blanchardb, BlueCanary9999, Bluerasberry, Blutred, Bobblewik, Bobo The Ninja, Bobo192, Bogey97, Bongwarrior, Bovineone, Braaropolis, Brian0918, Brianporter, Brick Thrower, Brothejr, BryanG, Bubba hotep, Bubba73, Bubzyz, Bucketsofg, BuickCenturyDriver, Bullzila, CTZMSC3, Caltas, Can't sleep, clown will eat me, CanadianLinuxUser, Canaima, CannedLizard, Capricorn42, CaptainVindaloo, Cardonnell, Cburnett, Cdc, CecilWard, Ceo, Cepopaladin, CesarB, Chacor, Chainz, Chamal N, Chaoserver, Chase me ladies, I'm the Cavalry, Chasingsol, Cherron1994, Chick Bowen, Cholmes75, Chopeen, Chris the speller, ChrisHodgesUK, Chriswiki, Chromaticity, Chuunen Baka, Chzz, CiTrusD, Cj005257-public, Cleared as filed, Cntras, Coasterlover1994, Cobi, Cometstyles, Comex, ConconJondor, Connormah, Controloye, Conversion script, Corpx, Corti, Corvus cornix, Courcelles, Crazycomputers, Crazytales, Creative0o, Cremepuff222, Crystallina, Cst17, Cureden, Curps, Cwolfsheep, Cybjit, Cynicism addict, D-Katana, DARTH SIDIOUS 2, DJ Clayworth, DMahalko, DVD R W, DaL33T, Dadude3320, Damicatz, Damore1405, Dan6hell66, Dandorid, Daniel Olsen, Daniel5127, DanielCD, Danny, Dante20XX, Dark MooGoo, Darrell Greenwood, Darth Panda, David Levy, David.Mestel, Davidoff, Dawn Bard, Dcljr, Ddas, DeadEyeArrow, Deagle AP, Deannyc, Deman 24, Demmy, Denelson83, DerHexer, Dharris, Digitalme, Dillard421, Dina, Diomidis Spinellis, Dionyziz, Discospinster, Dmaftei, Doc Daneeka, Doniago, Doria, Dougofborg, Doulos Christos, Dr.queso, Dragonball1986, Dragons flight, Dreadstar, Drkmaster, Drmies, Drunken Pirate, Dspradau, Dungodung, Dysepsion, Dzof, E0steven, ESkog, EagleOne, East718, Echuck215, Edward, Edward301, Egmontaz, El Monster, ElKevbo, Elassint, Elockid, Emersoni, Emo Elli, Emurphy42, Enviroboy, Epbr123, Er Komandante, Eric Kvaalen, Eric4, Escape Orbit, Evercat, Everyking, Excirial, Fagiolonero, Falcon8765, Famspear, Faradayplank, Farquaadhnchmn, FastLizard4, Favonian, Fdgsdfgg rgfsfdg fdsgsrfff, Feezo, FellowWikipedian, Ferrija1, Fieldday-sunday, Fleizach, Floaterfluss, Flyguy649, Forkazoo, Fox816, Fran Rogers, Francis22, FrancoGG, Frankie0607, Frap, Frazzydee, FreeKresge, FreplySpang, Freshmaniac, Friendly Neighbour, Fritzpoll, Frozenpandaman, Funky Monkey, Furrykef, Fuzheado, Fuzzypeg, Fvasconcellos, Fvw, G.A.S, GICodeWarrior, GPugh, GT5162, Gadfium, Gail, Gaius Cornelius, Galoubet, Gamaliel, Gamera2, Gary99129, Gatlingunlevel27, GcSwRhIc, Geeoharee, Gengiskanhg, Geoffrey, George The Dragon, George2001hi, Gerardkcohen, Giftlite, Gilliam, Glane23, Gnowor, Gogo Dodo, Golbez, Goldom, Gonzo fan2007, Goodnightmush, Gracefool, Graciella, GraemeL, Graham87, Greswik, Grim23, Grunt, Guardians611, Guest9999, Gurch, Gurchzilla, Gwalla, Gwernol, H8erade, Haakon, Hacii, Hadal, Hagerman, HalfShadow, Hall Monitor, Hansmohrid, HappyInGeneral, Haza-w, Hbackman, Hcberkowitz, Hdt83, Henry W. Schmitt, Heracles31, HiDrNick, Hmrox, Hojimachong, Home-4f8918a9a7.\mshome, Hon-3s-T, Howabout1, Hughcharlesparker, Hughesey, Huskihuskihuski, Hut 8.5, Huttarl, Hydrargyrum, II MusLiM HyBRiD II, IRP, Iancarter, Idcmp, Idts, Ilya, Imisslife, Immunize, Imperator3733, Imroy, Imveryveryveryverybored, InShaneee, Indeterminate, Infrogmation, Inspector 34, Intelati, Iorek85, Ipatrol, Iridescent, Irishguy, IronGargoyle, Iseeaboar, Ixfd64, J-stan, J.delanoy, J450NH3, JDoorjam, JForget, JHMM13, JLaTondre, JR JakeRs, JTN, Ja 62, JaGa, Jackfork, Jackohare, Jackol, Jake Nelson, James ONeal, JamesAM, Jaredbelch, Jasper Deng, Jauerback, Jaxl, Jebba, Jeepday, Jeff G., Jeffrey Mall, JeremyA, JesseW, Jh51681, JidGom, Jj137, Jklin, Jnc, JoanThaBone, JoanneB, JoeKearney, Joelr31, John of Reading, John254, JohnWittle, Johndarrington, JohnnyB256, Johnteslade, Johnuniq, Jojhutton, JonHarder, JorgeGG, Josh Parris, JoshSkidmore, Jossi, Joyous!, Jpatokal, Jsalims80, Jsc83, Judik, Juliancolton, Junckerg, Junnel, Jusdafax, Jusjih, Jwissick, K12345wiki, KFP, Kafka Liz, Kaisershatner, Kanonkas, Kaplin, Karada, Karl-Henner, Katalaveno, Katieh5584, Kbdank71, Kbrose, Kelly Martin, Kesac, Kevin B12, Kevin66, KevinTjan, Kgfleischmann, Khukri, King of Hearts, Kingjalis3, Kingpin13, Kinu, Kipoc, Kirill Lokshin, Kjd, Kjkolb, Klepas, Klose99, KnowledgeOfSelf, Kookykman, Koolkat2448, Kraftlos, Krellis, Krun, Kryptops, Kthsujal, Kungming2, Kuru, Kurykh, Kvng, Kwamikagami, Kwoksir, L'ecrivant, La Parka Your Car, La Pianista, LaMenta3, Labongo, Lam4o, Lanasa, Latka, Law, Lawrence Cohen, LeaveSleaves, LegitimateAndEvenCompelling, Les boys, LibLord, Lightdarkness, Lilac Soul, Lilserif, LinguistAtLarge, Litefantastic, Little Mountain 5, LittleOldMe, LizardJr8, Lokimang, Lord Voldemort, Loren.wilton, Lostintherush, Lothar von Richthofen, Lotje, Lucinos, Luke wyatt, Luna Santin, Lupin, LyonJE, MER-C, MSSEVER, Mabuse, Mac, MacroDaemon, Maddiekate, Magister Mathematicae, Magnus Manske, Makeemlighter, Malcolm Farmer, Malhonen, Malo, Mani1, ManiF, Manop, Marek69, Mark Renier, MarkSutton, Martarius, Martynas Patasius, Master Jay, Master of Puppets, Mastershake phd, Matthewrbowker, Matthuxtable, Mattv2006, Maurog, MaverickSolutions, Maximillion Pegasus, Maxis ftw, Maxweb, McSly, Melonite, Melter, Mentifisto, Merovingian, Metaeducation, Mets501, Michael Hardy, Michael Pheddyn, Mikaey, Mike Rosoft, Mindmatrix, Minna Sora no Shita, Mion, Misteror, Mld zero, Mmeerman, Mmernex, Mmmready, MonoAV, Montchav, Monty845, Moomoomoo, Moonriddengirl, Mormegil, Morning277, Mr.Z-man, MrJones, Mrsudip, Mschel, Mulad, Murderbike, Mushroom, Mwtoews, My Cat inn, Mygerardromance, Myhihihi, Myiptest, Myransree, Mário, NSLE, Nagy, Naive cynic, Nakon, Nanshu, Nascar1996, Nastajus, Natalie Erin, Nathanrdotcom, NawlinWiki, Neeples, NewEnglandYankee, Newwhist, NickBush24, Nicklous20, Nightscream, Nimiew, Ninny777, Nishkid64, Niteowlneils, Nivix, Nlu, Nn123645, Nneonneo, No Guru, Noctibus, Nofallah, Nrlight, Nsaa, Nubiatech, Obakeneko, Ocatecir, Odistery, OhmyΩ, Ohnoitsjamie, Oliver Lineham, OlivierMehani, OllieFury, Omicronpersei8, OnePt618, Onecountry, Opelio, Orange Suede Sofa, OrangeDog, Orangutan, OrganizeFISH, Oshra schwartz, Ottawa4ever, OverlordQ, Oxymoron83, PRRfan, Pabix, Pachydermballet, Paddu, Pandion auk, Pantjz, PatchesTheCaveman, Patpink, Patrick, Patstuart, Paul Carpenter, Paul Magnussen, Paul Stansifer, PaulHanson, Pavel Vozenilek, PedroPVZ, Pepper, Perilwalruz, Persian Poet Gal, Peruvianllama, Peter, Peyre, Pg2114, Pharaoh of the Wizards, Phatom87, Phgao, PhilKnight, Philip Trueman, Piano non troppo, PierreAbbat, Piet Delport, Pilotguy, PinchasC, Pinethicket, Pmlineditor, Pmsyyz, Pnm, Polaed, Ponies123456789, Poochy, PoopyPantsMcPooperson, PrestonH, Prolog, Pseudomonas, Psy guy, Puffin, Pyrop, Pyrospirit, Quintote, Quuxplusone, Qwyrxian, Qxz, RDaneel2, Rachtchi, Radiant chains, Radioactive afikomen, Rafasseb, RaiderTarheel, RainbowOfLight, Ralphwiggam75, Rameshbabu.itian, Rantsroamer, RattleMan, Razakausar, RazorICE, Rcawsey, Rcmarotz, Rdsmith4, Reconsider the static, RedHillian, RedWolf, Regancy42, RenRochefort, Rettetast, RevolverOcelotX, RexNL, Rfc1394, RiK harhar, Riana, Rianvisser, Ricgal, Rich Farmbrough, Richard001, Rjensen, Rjstott, Rlcantwell, Rmt2m, Rmustar 10, Robbe, Robby.is.on, Robocoder, RockMFR, Romanskolduns, Ronhjones, Rory096, RoyBoy, Rrburke, Rror, Rsm99833, RunOrDie, Ryamigo, Ryt, SEWilco, SGBailey, Sahansuraweera, Salamurai, Samilamethmal, Sango123, Santamage98, Sasquatch, Sayden, Scgtrp, Schzmo, ScienceGolfFanatic, Sean D Martin, Sean.hoyland, Seanx820, Sebastian Goll, Sebleblanc, Sgarcia05, Shadowjams, Shadowlynk, Shanel, Shanes, Sherool, Shiftoften66, Shirik, Shirulashem, Shiva 29, Shoeofdeath, Shotwell, SidP, Sietse Snel, Sillicongal, Simetrical, Simon J Kissane, SimonP, Simonwheatley, Simple Bob, Sims2789, Simsong, SingingDragon, Sir Nicholas de Mimsy-Porpington, Sjakkalle, Sjö, Skier Dude, Skunkboy74, SlimVirgin, Slon02, SluggoOne, SmallPotatoes, Smileyrepublic, SmilingBoy, SoSaysChappy, Soliloquial, SonicAD, Soumyasch, Soundcomm, SpK, Spazit, Spazturtle, SpeedyGonsales, Spellcast, Sportsfan 555, Spundun, SpuriousQ, SqueakBox, SquidSK, StabiloBoss, Stephenb, Superm401, Superuserit, Supreme Deliciousness, Suruena, SusanLesch, Swalot, SymlynX, Synchrite, T3h 1337 b0y, TJ Spyke, TYelliot, Tangotango, Tannin, Tanthalas39, Tanweer Morshed, Tao, Taw, Tbhotch, Tcncv, Ted Longstaffe, Teh tennisman, Teles, Tgotchi, The Anome, The High Fin Sperm Whale, The Return Of Squad, The Rogue Penguin, The Thing That Should Not Be, TheCoffee, TheGrimReaper NS, TheKMan, Thecosmos, Thegraham, Thekittenofterra, Thewayforward, Thingg, Threeafterthree, Thue, Tide rolls, TigerShark, Tim1988, Timpailthorpe, Titoxd, Tjwagner, Tom harrison, Tommy2010, Tothwolf, Towel401, Tree Biting Conspiracy, Tresiden, Troy 07, Trusilver, Trysudhakar, Ttlkr, Tverbeek, Tvoz, Twinkiesarethesourceofallknowledge, Twinsday, Tyw7, UFu, Ugur Basak, Ukexpat, Ulric1313, Uncle Dick, UncleBubba, UnknownzD, Uranographer, Useight, V0rt3x, VAcharon, Valhallia, VampWillow, Vancouverguy, Vandalismterminator, Vanished User 1004, Vary, VasilievVV, Vclortho, Veesicle, VegaDark, Vendettax, Versageek, Versus22, Vicki Rosenzweig, VirtuousCircle, Viscious81, Vrenator, Vrraybadboy78, VzjrZ, W, WJBscribe, WadeSimMiser, Wavelength, Wayne Slam, WaysToEscape, Wayward, Weedwhacker128, Wengero, Wertuose, Weyes, Who then was a gentleman?, Whomp, Why Not A Duck, Wiigocrazii, Wikedit-34, Wiki alf, Wiki104, WikiLaurent, Wikibofh, Wikifan07, Wikipelli, William Avery, Willking1979, Willysmilk, Wilson Tam, Wimt, Wisco, Wmahan, Woody, Wrs1864, Wysprgr2005, Xagent86, Xezbeth, Yahel Guhan, Yamamoto Ichiro, Yamla, Yanksox, Yelyos, Yidisheryid, Yug, Zachlipton, Zarcillo, Zchenyu, ZeWrestler, Zeimusu, Zemadmat, ZeroOne, Zhackwyatt, Zocky, Zondor, Zsinj, Zzuuzz, شات صوتي, පසිඳු කාවින්ද, 2790 anonymous edits

IP forwarding algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=452353026 *Contributors*: Ka843517, Malcolmxl5, MatthewVanitas, Ngriffeth, 3 anonymous edits

IP fragmentation *Source*: http://en.wikipedia.org/w/index.php?oldid=454918961 *Contributors*: Anandvel, Anonauthor, Bentogoa, Brainsik, Conrad.Irwin, Dandorid, JTN, JamesHDavenport, Kbrose, Logan GBA, Loniousmonk, Mild Bill Hiccup, Minimac, N328KF, Nealcardwell, PeterEasthope, Plugwash, Pufferfish101, Queenmomcat, RedWolf, Woohookitty, Wpiechowski, Wrs1864, 17 anonymous edits

IP in IP *Source*: http://en.wikipedia.org/w/index.php?oldid=456730725 *Contributors*: Andvd, Coolcoder, Cybjit, Malcolma, Rjwilmsi, SMasters, 4 anonymous edits

IPv4 *Source*: http://en.wikipedia.org/w/index.php?oldid=463895389 *Contributors*: -Majestic-, A.R., Abdull, Acather96, Adrian.benko, AlephGamma, Alexkon, AlistairMcMillan, AndreasWittenstein, Andrewmc123, Andypar, Angela, Anna, ArglebargleIV, Armando, Axelriv, Barro, Barryd815, Begoon, Bmpercy, Borgx, Brest, Bro1960, Brouhaha, C. A. Russell, CCFreak2K, CWii, Calcwatch, CaribDigita, Carlo.arenas, Cburnett, CecilWard, CesarB, Chris D Heath, Chrishmt0423, Christan80, Cm115, Cmichael, Conversion script, Coredesat, Corti, Crashdoom, Cwolfsheep, Cynical, DARTH SIDIOUS 2, DH85868993, Dan6hell66, Daniel Staal, Danielbarnabas, DataWraith, DenisKrivosheev, Dmaftei, Dnas, Dotshuai, DreamGuy, Dsearls,

Duffman, Dunganb, Dungkal, Ed g2s, EdC, Ekspiulo, El C, Electron9, Enjoi4586, Ericbarch, ErikWarmelink, Exallium, Faco, Floydpink, Formulax, Fred Bradstadt, Fredrik, Fresheneesz, Gallando, Gehlers, General Wesc, Giftlite, Glrx, Graciella, Graham87, Graven69, Hairy Dude, Hcberkowitz, ILike2BeAnonymous, Ilario, Imroy, Indrek, Ironholds, JTN, Jasper Deng, Jbergste, Jbohac, JeroenMassar, Jgeer, Jk2q3jrklse, Jnc, Joelby, Johnnyboyshoots, Johnteslade, Joseph Solis in Australia, Joshua, Jpyeron, Kaal, Karada, Kasperl, Katieh5584, Kbrose, Kevin66, Kgfleischmann, Khr0n0s, Kickboy, Kiore, Kmwiki, Krellis, Kvng, Kwamikagami, Kyng, Leuqarte, Liface, Lightmouse, Lph, Lukeritchie, MECU, Magioladitis, Magnus.de, Maimai009, Mange01, Marcoscm, Markrod, Melnakeeb, Mewashere1, MichaelGoldshteyn, Milan Keršláger, Mindmatrix, Miss Saff, Mmtrebuchet, Mochi, Mokgen, Molerat, Morten, Mushroom, Nathan Hamblen, Nealmcb, Neelix, NewEnglandYankee, Nil Einne, Noldoaran, Nubiatech, Ojw, Omniplex, Onceler, Opelio, Outback the koala, Parent5446, Parkamark, Paul, PaulHanson, Pengo, Peterhoneyman, Philip Trueman, Phoenix314, Phorque, Piano non troppo, Pmj, Presto8, Ptmc2112, Raanoo, Ranto, Rantsroamer, Rblhjm, Rchandra, RevRagnarok, RexNL, Robert Brockway, RoySmith, Rpwoodbu, Ruwolf, RxS, RyanWKeen, Sarafankit, Scjessey, Seaphoto, Shane kerr, Simon J Kissane, Sirmelle, SmilingBoy, Smurrayinchester, SpacemanSpiff, Spearhead, SpectatorRah, SpeedyGonsales, Stephan Leeds, StrangerInParadise, Suruena, Swellesley, Taestell, Teemuk, The Anome, The Thing That Should Not Be, TheGreyArea, Themonstergila, Thnidu, Tide rolls, Toolnut, Tyler.szabo, UU, Ultimus, Undeference, Versus22, Visiting1, Vivio Testarossa, Wolfsbane2k, Woohookitty, Wrs1864, Wyksztalcioch, XavierHager, Xibe, Yyy, Zanetu, Zetawoof, Zfr, ^demon, 464 ,ירח אפל, anonymous edits

IPv4 subnetting reference *Source*: http://en.wikipedia.org/w/index.php?oldid=464789748 *Contributors*: Abdull, Alvestrand, Angela, Armando, AubreyEllenShomo, Bradyadams, Canaima, Cburnett, Charon, Crimson30, El C, EvilCouch, Frap, JTN, Jclemens, Jeepday, Jengelh, Jnc, JonHarder, Karada, Kbrose, Kl2217, Klaver, Momo san, Mtu, Patrick, Phoenixxxxxx, Raeilog, Rayrob, RedWolf, Reinderien, Saxbryn, Sillicongal, Stoni, Tenretnieht, The Anome, Wrs1864, 22 anonymous edits

IPv6 *Source*: http://en.wikipedia.org/w/index.php?oldid=464603641 *Contributors*: 09 F9 11 02 9D 74 E3 5B D8 41 56 C5 63 56 88 C0 - hack, 128.107.253.xxx, 2001:db8, 2345us, 49oxen, A3 nm, AJR, Abune, Accountholder, AceJohnny, Adam Conover, Adam Krellenstein, Adamthewebman, Adoniscik, Aeluwas, Aeons, Agent00111, Ahoerstemeier, Aigarius, Ajbool, Akamad, Alainkaa, Alansohn, Aldie, AlephGamma, Alex43223, Alexander UA, Alexkon, Alexwcovington, AlistairMcMillan, Allanlw, Amigan, Amybajwa, AnandKumria, Anastrophe, Andareed, Anders Feder, Andonic, Andrei Stroe, Andrew4u, AndyTheGrump, Annesville, AnonMoos, Antonis Christofides, Arbitrary username, Arichnad, ArnoldReinhold, Artemgy, Arthur Rubin, Aschwegmann, Ashley Y, Astronautics, Atakdoug, Auntof6, Avanu, Axelriv, Baa, Badcalculon, Balachanderk, Barri, Bazsa, Bbpen, Bdesham, Belamp, Beland, Benabik, Benandorsqueaks, Benbest, Bender235, Bentendo24, Beoba, Bertra g, Bgraabek, Big Brother 1984, BioTube, Blaynew, Blubber42, Bobblewik, Bobo192, Boothy m, Borgx, Bornapilot, Bousquf, Bovineone, Brianmaddox, BrokenSegue, Brownsteve, Bruce404, Bsv109, Bugfood, BurnDownBabylon, CKD, CKerr1, CableCat, Calvin 1998, Cameltrader, Cananian, Cap'n Refsmmat, Capek, Caper13, CapitalR, Captkirk, Carlosguitar, Centrx, CesarB, Cesarolvera, Cgdallen, ChaosR, Charliel 94, Charu, Charwinger21, Chaser, Chbarts, Chris 73, ChrisErbach, Chrisinspfld, Cinnamondouche, ClamDip, Closedmouth, Cmdrjameson, Cmsb705, Coaxial, Cobi, Cometstyles, ComputinChuck, Confuciou, Conversion script, Corso84, Crazycomputers, Crd Alameda, Credema, Crispmuncher, Crwth, Cst17, Cwolfsheep, Cyan, Cybercobra, Cybjit, Cyp, DARTH SIDIOUS 2, DBooth, DMahalko, DNSstuff, Daev, DalZot, Dale Arnett, Dan100, Dandorid, Daniel Luechtefeld, DanielDeGraaf, Darguz Parsilvan, Das7002, DataMatrix, Dave6, David Gerard, Davidhorman, Dawnseeker2000, Deepeedoyle, Defyant, DerekMorr, Dglynch, Diannaa, Dickguertin, Dicklyon, Dispenser, Djschaap, Dlabtot, Dlange13, Dnas, Dnevil, DocKrin, Dolda2000, Don4of4, Donreed, Dontopenyoureyes, Drangon, Drearwig, Drewheasman, Droob, Dwmalone, Dylan Lake, Dylan620, E94mli, EDUBLE, EdBever, EdC, EdoDodo, Edokter, Edward, Ehn, El C, Eldar, Emonk72, Endpoint, Enjoi4586, Enquire, Eradt11, Erc, EthanL, EugeneZelenko, Euphrosyne, Evil Monkey, Extropian314, FGont, FT2, Falcon8765, Falcon9x5, Farncombe, FatalError, Feedmecereal, Feureau, Fewaffles, Figz, Firealwaysworks, FlieGerFaUstMe262, Fluffy 543, Fnagaton, Folajimi, Foobar, Fragglet, Frap, Fredrik, Freeaqingme, Fresheneesz, Frnkblk, Fsterry, Galmicmi, Geekening, GerardM, Ghane, Giftlite, Glane23, Glen Hein, Glenn burdett, Glenndavies, Glrx, Gmaxwell, Gmedding, Gorffy, GovStuff, Graciella, Graham87, GrahamHardy, GreenReaper, Greg L, Gudeldar, Guiltyspark, Gunnala, Gurch, Gwernol, HAl, Haakon, Hairy Dude, Haleyga, Ham Pastrami, Hankwang, Hannes.nz, HarryHenryGebel, Hashar, Hatu7, Hawaiian717, Hcberkowitz, Hdt83, Helix84, Hfastedge, Hires an editor, HobbesLeviathan, Holizz, Holmwood, HorsePunchKid, Hpa, Husky, Hvn0413, Hypersonic, II MusLiM HyBRiD II, Ihope127, Ilja Lorek, Iluvcapra, Imarsman, Imroy, Imzogelmo, Incnis Mrsi, Infofarmer, Insanity Incarnate, Int21h, InterMa, Intgr, Intrepion, Iromeister, Ironholds, Itojun, Ivolocy, J-D-Cronin, J128, JHunterJ, JTN, Jaizovic, Jamesday, JameySharp, Jan1nad, Jansenguus, Janto, Jarry1250, Jasper Deng, Jclemens, Jddahl, Jec, Jeff G., Jeff02, JeffMorriss, Jeffsw6, Jengelh, Jeremy Visser, JeroenMassar, Jfromcanada, Jhd, Jhwoodyatt, Jithrae, Jj137, Jnc, Jo3sampl, JodyB, JoeKearney, John Maynard Friedman, JohnOwens, Johnuniq, JonDePlume, Jordandanford, Jordipalet, Joshua Scott, Jtg, Julesd, Justjohn45, K0rana, KDS4444, Kaldari, Karada, Kasperd, Kattmamma, Kaypoh, Kb966k, Kbolino, Kbrose, Kcordina, Keilana, KelleyCook, Kenyon, Kestasjk, Kgfleischmann, Kieff, Kinema, Klaver, Koavf, Konikofi, Kozuch, KrakatoaKatie, Krellis, Kroneage, Kthomas8, Kusma, Kvaks, Kvng, Kynan, LUUSAP, LVXN112, Lagesag, LakeHMM, Latifladid, Laug, Lawrence Cohen, Lightbulbcole, Likestheaction, Logan, Lord Chamberlain, the Renowned, Lotje, Lousyd, Luigiacruz, M. B., Jr., M00dawg, Maalaoui, Macrakis, Madhav208, Madman91, Maduskis, Mahtin, Maian, Majordragon, Manankanchu, Mange01, Mansoor.riz, Marcod'Itri, Marius p, Mark Bergsma, MarkMLl, Markus Kuhn, MarkusWanner, Martarius, Martyvis, Marudubshinki, Matchups, Mathboy965, MaxWilder, Maxim Masiutin, Mdd4696, Meand, Megan at ARIN, Meneth, Mentifisto, Mhd.ashraf, Michael B. Trausch, Michael miceli, Michael.davie, MichaelBillington, Michaelrayw2, Midnightcomm, MihalOrela, Mike Rosoft, Mike Schwartz, MikeWren, Mikm, Mindmatrix, Minghong, Mintleaf, Mitar, MithrasPriest, Mmxx, Mobus, Molerat, Money23, Mordomo, Mormonsareloser, Mr Minchin, MrOllie, Mro, Msiebuhr, Muad, Mukkakukaku, Mulad, MureninC, N328KF, NYMets2000, Nacnud22032, Nageh, NameIsRon, Napalm Llama, NapoliRoma, Narellec, Nasa-verve, Nate Silva, Naveenpf, Nealmcb, NerdBoy1392, Netrangerrr, Nhandler, Nicd, Ninjagecko, Nixdorf, Nixeagle, No More TV, Noldoaran, Northgrove, Notbyworks, Nsaa, Nubiatech, Nurg, Nwbeeson, Nzseries1, OSborn, Od Mishehu, Ohnoitsjamie, Olipro, OlivierMehani, Oliwan, Omegatron, Omegium, Omicronpersei8, Omniplex, Oneliner, Orchistro, Ost316, Ovideon, P.vishnu7, PabloStraub, Paine, Pankkake, Paolopal, Paradox2, Paranoid, Parsmutaf, Pascal666, PassportDude, Pathoschild, Patrickdavidson, Paul1337, PaulHanson, Peak, PedroPVZ, Pekaje, Pengo, Personman, PeterCScott, PeterKz, PeterStJohn, PhilHibbs, Philc 0780, Phoe6, Phoenix-forgotten, PierreAbbat, Pinkgothic, Pjrm, Planetscared, Plasticup, Plau, Pnm, Pointillist, Pontillo, Powerlord, Prabhuhwiki, Prolog, Prunesqualer, Psheld, Psz, Ptmc2112, Public Menace, Puchiko, PuerExMachina, Quaestor23, Quantumobserver, Quebec99, Qwertytech, Qwertyus, RHaworth, Raffen, Raghith, Rait, Ralphcase, Random832, Rchandra, Rcloran, Rd232, Rdenis, RedWolf, Redlazer, Rememberway, ReyBrujo, Reza 2638, Rgaushell, Rich Farmbrough, Richard W.M. Jones, RichiH, Rick Sidwell, RickBeton, Rilesman, Rischmueller, Roadrunner7, Robaire5006, Robert Brockway, Robomaeyhem, Rocketman768, Ronz, Roy.wonder.cohen, RoyBoy, RoySmith, Rps, Rrius, Rwessel, Ryankearney, Rythie, Ryulong, Ryuzaki00, ST47, SWAdair, Sam Hocevar, Samantha of Cardyke, Sandeepsoman, Saran945, Sarutv, SaveTheRbtz, Scientus, Scj2315, Scott.somohano, Scottbadman, Scratchy, Seano1, Sebcastle, Sega381, SelfStudyBuddy, Setherson, Sgeeves, Shadowjams, ShakataGaNai, Shirishag75, Shultz IV, Sibi antony, Sick bug, Sigkill, SillyWilly, Simon J Kissane, Sirmelle, Sjmsteffann, Sjrosen, Skittleys, Skybon, Smallpond, Smarter1, SmilingBoy, Smurfy, Smurrayinchester, Snaxe920, Snorgy, Snottywong, Sobec, Socialservice, SolarWind, Some jerk on the Internet, Sonicsuns, Sosodank, South Bay, Southen, Spearhead, Spicemines, SpuriousQ, Squids and Chips, Stephan Leeds, Stephenb, SteveDavidsen, StuartBrady, Stux, Stwalkerster, Sukenjain, Suruena, Suseno, Svick, SymlynX, Synchrite, Sysy, TJJFV, TJRC, TankMiche, Team4Technologies, Teemu Maki, Teemuk, Template namespace initialisation script, Tenth Plague, Terillius, Terjepetersen, Tfl, ThG, The Anome, The Last Username I Could Think Of, The Nut, TheAnarcat, TheLight, Theant2000, Thehotelambush, Thewallowmaker, Thexchair, Thorpe, Thrindel, Thumperward, ThurnerRupert, Tim Capps, Titoxd, Tmaufer, Tmh, To Serve Man, Tommy2010, Towel401, Trejrco, Trevor Johns, Tristanb, Trisweb, Trusilver, TwoHundredOk, Tylerl, Ukh, Uncle Milty, UncleDouggie, Unclevinny, Und1sk0, Unitacx, Universalcosmos, Utility Knife, Vajrallan, Valenciano, Vannessa.beth, Vedantm, Vegaswikian, Verdy p, Veriodbg, Vespristiano, Victor, Viniciustinti, Viperious, Visiting1, Voidxor, Voomoo, WJBscribe, Wahjava, Warren, WarthogDemon, Watain, Wavelength, WayneMokane, Weatherman1126, Weregerbil, Wetman, Weyes, Whatisipv6, Wikimoder, WikipedianYknOK, Wimt, Wisco, Wk muriithi, Wmahan, Wonderstruck, Wrs1864, Ww, Wwoods, X-Fi6, Xandell, Xeltran, Xerces8, Xpclient, Yago, Yama, Yayay, Youssefsan, Yurik, Yurivict, Zachlipton, Zaphraud, Zed5linux, Zeerak88, Zemyla, Zhackwyatt, Zmding, ZorphDark, Обедающий философ, 1573 anonymous edits

IPv6 packet *Source*: http://en.wikipedia.org/w/index.php?oldid=458639827 *Contributors*: Closedmouth, Dandorid, Danhash, Eraserhead1, Jec, Kbrose, Kvng, LilHelpa, Mr. Wheely Guy, Mro, Muhandes, Nageh, Nasa-verve, Nick Number, Rwessel, TheRa'ike, Weylin.piegorsch, Woohookitty, 12 anonymous edits

Maximum transmission unit *Source*: http://en.wikipedia.org/w/index.php?oldid=461881430 *Contributors*: 213.253.39.xxx, 217.125.126.xxx, 802geek, Abdull, Adambiswanger1, Aldaron, Ali@gwc.org.uk, Andrewborrell, Babbage, BenFrantzDale, Betbest1, Borgx, Btilm, CIreland, CecilWard, Chameleons84, Conversion script, CosineKitty, Cwolfsheep, Darth Panda, Dhammala, Dkleeman, Dnas, DylanW, EagleOne, El Cubano, Enduser32, Epbr123, Fleminra, Ground Zero, Hamster2.0, Hgmichna, I2so4, Isnow, Itusg15q4user, J.smith, JTN, Jandalhandler, Jasrocks, JeLuF, Jkl, JohnOwens, JonHarder, Julesd, Kaare, Kenyon, Kvng, MOM4Evr, MS3FGX, ManuelGR, Mboverload, Milan Keršláger, Mormegil, N3ddy, Niclas Wiberg, Nightstallion, Pedant17, PeteShanosky, Peyre, Piet Delport, Plugwash, Prondou, Rich Farmbrough, Richard W.M. Jones, Ricojonah, Rjwilmsi, Rvalles, SCΛRECROW, Shaddack, Skyschulz, Suplamer, Tempshill, Tgwaltz, Thumperward, Tobias Bergemann, WaffleMonster, Waskage, WereSpielChequers, Whitepaw, Winterst, X!, Xaje, Xcentaur, Xchbla423, Xxsquishyxx, Zaf, ZoFreX, 176 anonymous edits

Mobile IP *Source*: http://en.wikipedia.org/w/index.php?oldid=465147490 *Contributors*: Adrignola, Aleichem, Alexius08, Alimedany, Altsheikh1, Arkrishna, Audriusa, Bjanders, Bruce1ee, Chrislk02, Danielcohn, Darossetti, Defender of torch, Doprendek, Epbr123, Frap, Fresheneesz, Goarany, Goldom, Infindebula, Isj-wikipedia, J. Spencer, Jeffrey Mall, Jengelh, Johnuniq, JonHarder, Kbrose, Kevinmon, Kinema, Koshoid, Kulin shah, Liangent, Lionking timone, M7amad4, Magioladitis, Mandarax, Mange01, Maryhit, MathsPoetry, Mgarlop, Mgaved, Michael Hardy, Mintleaf, Mlaffs, Mustafaerg, Netrangerrr, Nneuman, Nono64, NortyNort, Oli Filth, Pearle, Plustgarten, Porttikivi, Radiant!, Ranjithsutari, Rchandra, Rdcole, RedWolf, Rkrikorian, Ronhjones, Sandrewh, Schlauf, Sepper, Shadowjams, Shirik, Sietse Snel, Sjlver, Starbois, TelecomNut, TrogdorPolitiks, UninvitedCompany, Vahid talebi, Wequiry4278, Willem, Woohookitty, Wrs1864, Xrgtn, Zil, 139 anonymous edits

Multicast *Source*: http://en.wikipedia.org/w/index.php?oldid=461689808 *Contributors*: A.k.a., A234atA234Workshop, Alano, Aldie, Arunsjamwal, Awcolley, Babbage, BenMosher, Bomazi, Bomberharris, Borgx, Bovineone, Cbdorsett, Cgbraschi, Chris the speller, Christian.pena, Chriswaterguy, Compustrat, ContentCreator, Conversion script, Cpell, Ctempleton3, Cyr, Damian Yerrick, Darth Panda, Dcsohl, EagleOne, Easyas12c, El C, Epah, ErkDemon, Face, Feb30th1712, Fordsfords, Frogbegi, Glenn, Gracefool, GrammarNSpellChecker, Hairy Dude, Isnow, Itai, JLaTondre, JTN, Jamelan, Jamoche, Javifs, JidGom, Jim.henderson, Jleedev, Johnuniq, JonHarder, Jonathan de Boyne Pollard, JordoCo, Jtk, Kbrose, Kheldysh, Kmag, Kooo, Krant, Kurt Shaped Box, Kvng, Kyng, M4gnum0n, MONGO, Malishav, Maltest, Mammutti, Markus Matern, Mdavids, Mendel, Mhardee, Miami33139, Michael Hardy, Mindmatrix, Minna Sora no Shita, Mononomic, Mortein, Mrh30, Mulad, Multi io, Natano, Nburden, Neilc, Nimiew, Nimur, Nityanandparab, Nixeagle, Nubiatech, Nuno Tavares, Oscarthecat, P0lyglut, PabloStraub, Peter Karlsen, Petergwilson, Plugwash, Porceberkeley, Proski, Raanoo, Radiosband, Rich Farmbrough, RoyBoy, Rpdner, Rrburke, Sae1962, Sbisolo, SethFreeman, Setver, Shanejast22, Sigkill, SlubGlub, Socialservice, Spearhead, Stardancer, Stephen.frede, Superm401, Suresh creation, Suruena, Svick, SymlynX, The Belgain, TheMandarin, Tinucherian, Tonkie, Unyoyega, Vvim, Wavelength, Wbrameld, Wearealmosthere, Wiggin15, Will Beback, Woohookitty, Yaronf, Ybbor, Zoobee79, Zxb, 214 anonymous edits

Multicast address *Source*: http://en.wikipedia.org/w/index.php?oldid=458047453 *Contributors*: Aldie, Arimus, Awcolley, Brianhe, Carlo.arenas, Dandorid, DennisIsMe, Enkeedou, Eqvinox, FJPB, Fisherrider, Frap, Gardar Rurak, Gazoduc, Hairy Dude, Inightmare, JTN, Jez9999, Johnteslade, JonHarder, Jonathone, Jpcarter, Kbrose, Kms, Kooo, Kvng, Lexein, Logan GBA, Mikeblas, Mincebert, Mro, Muhandes, Mulad, Networksecurityguy, Niteowlneils, Nkaliape, PS2801, Raagumd, RedWolf, Rexfujikawa, SixSix, Sosodank, Squater, Stenn, Stephan Leeds, Suruena, Takanoha, TheWickerMan, Thoobik, Tristanb, Twitsed, Yayay, Zundark, 86 anonymous edits

Multihoming *Source*: http://en.wikipedia.org/w/index.php?oldid=460851927 *Contributors*: Beland, CyborgTosser, Demus Wiesbaden, Folajimi, Frap, Frencheigh, FreplySpang, Gracefool, Greg Ward, Haya shiloh, InfoEcon, Inkling, JTN, Josh Parris, Jphebert, Kbrose, Kwi, Mange01, MarieLG, Maximus Rex, Michael B. Trausch, Moink, Mozzerati, Mro, Nikolas Stephan, Roland Abt, Springnuts, Stefan Bethke, The Anome, Tide rolls, Tomcat66 g500, Tomchiukc, Tylerl, Woohookitty, 77 anonymous edits

Network address translation *Source*: http://en.wikipedia.org/w/index.php?oldid=462398046 *Contributors*: (, 65.29.90.xxx, Aapo Laitinen, Adam Krellenstein, Aelantha, Aitias, Ajo Mama, Alan U. Kennington, Alansohn, Aldie, Alex Smotrov, Alex.atkins, Alex.zeffertt, Alexhixon, AlistairMcMillan, Althena, Altrn8r, Andrew Hampe, Andrewpmk, Andrewriddell2, Aneah, Angela, Ap, ArsénireDeGallium, Ashwin, Asymmetric, Balajisarathi, Bbpen, Benoit rigaut, Bevo, Bos-Herz edit acct, Brion VIBBER, Brynosaurus, Cate, Cbarbry, Cburnett, CesarB, Cf. Hay, Cheung1303, Chowbok, Christian75, Conversion script, Copsewood, Cotoco, Crazycomputers, Crispmuncher, CrucifiedChrist, CyberSkull, Cybjit, D235, DARTH SIDIOUS 2, Daf, DanielEng, Daveg1k, Daveofthenewcity, Dawnseeker2000, Dcoetzee, DevastatorIIC, Dgtsyb, DiGiT, DisillusionedBitterAndKnackered, Droob, Drpixie, Drumzandspace2000, Dspradau, Dysprosia, EH74DK, Edcolins, Eddy264, Edward, Eolsson, Equendil, Ergy, Everyking, Evil Monkey, Excirial, Felipe1982, Fenix*NBK*, Fresheneesz, Gandaliter, Gareth Owen, Gary King, Garyvdm, Giftlite, Giraffedata, Glenn, Goatasaur, Golbez, GorillaWarfare, Gracefool, Graham87, Grimmfarmer, Guiltyspark, Guitargod2323, Hairy Dude, HarlandQPitt, Harrymcfogs, Hcberkowitz, Helix84, Hovden, Hydrargyrum, Icairns, Imcdnzl, Indrian, Iranway, Ivan Pozdeev, Ivan.Lt, J.delanoy, JHunterJ, JTN, Jan Kunder, Jasper Deng, Jengelh, Jez9999, Jhbdel, JidGom, Jokerspuppet, JonDePlume, JonHarder, Jondel, Jonshea, Josh Parris, Joshf, Joy, Jpbowen, Jsnx, Just Another Dan, Jyoti.mickey, KD5TVI, Karada, Karstbj, Kbdank71, Kbrose, Kenyon, Keycard, Kgfleischmann, Kristof vt, Kvng, Kwi, Kzollman, Lavenderbunny, Leuk he, LiDaobing, Lightdarkness, LittleOldMe, LobStoR, Lspo99, M gol, MARQUIS111, MER-C, Magnus Manske, Mallow40, Maltest, Mandarax, Mannafredo, Manop, Marchash, MarcoTolo, Mav, Mditto, Mercury543210, Mindmatrix, Mintguy, Misza13, Mmmeg, Murjek, Mygerardromance, Naniwako, Nazli, Nealmcb, Nilmerg, Nimiew, Nixdorf, Nubiatech, Nurg, Nyttend, Oalbacha, Oystein, PPBlais, Para, Pash, Pde, Pdelong, Peyre, Phatom87, Phenry, Philadams, Philbert2.71828, Piano non troppo, PierreAbbat, Pinkadelica, Plugwash, Pmsyyz, Pol098, Profchakraborty iitkanpur, Psychocim, Quarl, Rabarberski, Ramsey585, Rchandra, Rebecca, RedWolf, Rick Sidwell, Robert Brockway, Rohitthakral, Ross Fraser, Rrburke, Rushtoshankar, Ryan Roos, SF007, SLATE, SQL, SalineBrain, SaulPerdomo, Sbmehta, Seikku Kaita, Shahid789, Shirimasen, Shiro jdn, Sideswipe091976, Siipikarja, Simetrical, Simon South, SimonEast, SimonSellick, Slackerhobo, Smalljim, SoWhy, Sollosonic, Steelmans1980, Stephan Leeds, Stephenb, Steven Zhang, Sujirou, Svetovid, Syndicate, Taestell, Tagishsimon, TakuyaMurata, Teles, The Anome, The Inedible Bulk, Tiddly Tom, Tide rolls, Tobias Bergemann, Tommy2010, TonyHagale, Tresiden, Tristanb, Truthanado, Tsunanet, Tverbeek, Twilsonb, Ulric1313, UncleBubba, Urhixidur, VanishingUser, Wernher, Wiki alf, Wikipelli, Wimblykit, Winterheart, WithGLEE, WojPob, Wolf0403, Wolfkeeper, Wolfrock, Woohookitty, Wrs1864, Xmm0, Xpanzion, Yk4ever, YordanGeorgiev, Zap Rowsdower, Zhlmmc, Zondor, Zundark, 594 anonymous edits

Network interface controller *Source*: http://en.wikipedia.org/w/index.php?oldid=457423145 *Contributors*: 1005a055, 28421u2232nfenfcenc, A More Perfect Onion, A.macleay, AJNeblett, AThing, Aapo Laitinen, Aardnavark, Aboybap, Acroterion, Ageekgal, Alansohn, Amillar, An-chan, AndrewJSteele, Anomie, B.Zsolt, Blueskies238, Bobblewik, Boothy443, Bunnyhop11, Camw, Cathlazg, Chaosfinity, Cheesecheesewanttoplease, ChrisJMoor, Closedmouth, Cloudguitar, Codetiger, Color probe, Costello, Crazycomputers, DJ MeXsTa, DanielDeibler, Dawnseeker2000, DeadEyeArrow, Defireman, Delirium, Devesh.padmanabhan, Dia^, Diverman, Drae, EagleOne, Editore99, Ejay, Electron9, Ezeu, Fahadsadah, Felyza, Firetrap9254, Flamelai, Floquenbeam, Fluffernutter, FlyingToaster, Fox2030, Gadfium, Gf uip, Giftlite, Gona.eu, Grafen, Grim23, Guy Harris, Gzcsy3, Ham Pastrami, Hassocks5489, Hede2000, Helix84, Hornlitz, Hyeahmfr, IW.HG, Ikiroid, InsanityBringer, Intgr, Isaac Rabinovitch, IvanLanin, J.delanoy, JRaspass, JackTinWNY, Jackaranga, Jandalhandler, Jcbarr, Jdm, Jerusal, Jpgordon, Kankachi1980Kan, Kozuch, Ksg, Kvng, LOL, LyXX, Maelwys, Manassehkatz, Marc Mongenet, Margin1522, Marius, Materialscientist, Matty2002, Mboverload, Meekywiki, Mohamed Magdy, Moriori, Mxn, N5iln, Najle 13083, Natovr, NewEnglandYankee, Nillerdk, Nixdorf, Norm, Npthree, Nuno Tavares, Oliver202, Oxymoron83, P996911turbo, PJ, Philip Trueman, Piano non troppo, Pinethicket, Pinkadelica, PlanetCoder, PleaseStand, Pmsyyz, Quaeler, Qxz, Raf1qu3, Rancher 42, Removebeforeflight, Rhobite, Ricochet2200, Ringbang, Russob, Ruud Koot, S3raph1m, Samuella, Scohoust, Seaphoto, SebastianHelm, Sfoskett, ShelfSkewed, SineWave, Sjakkalle, Summercold143, Suruena, Swiftee18, THEN WHO WAS PHONE?, Taphilo, Tawker, Tbutzon, Tcbburleigh, The Thing That Should Not Be, TheJosh, Thumperward, Tim Yagan, Tlroche, Toyotaboy95, Travelbird, Treyt021, Turkishmonkey, UncleDouggie, Unyoyega, Utcursch, Vickey2020, Wayward, Wbenton, Wtshymanski, Yyy, עדיהל, كاشف عقيل, 395 anonymous edits

Network layer *Source*: http://en.wikipedia.org/w/index.php?oldid=461007512 *Contributors*: Abhi 4442003, Adrian M. H., Alfio, Arastcp, Arnavchaudhary, Arunachalammanohar, Ashishindemand, Bendykst, Borgx, Butko, Capricorn42, Colin Marquardt, Crawdaddio, Dgtsyb, Dicklyon, Digisus, Dominio, Dribbleboy, Dthomsen8, Egret, ElKevbo, EnOreg, Enchanter, Enjoi4586, Enochlau, Epbr123, Feezo, Fredrik, Giftlite, Goatasaur, Gouldja, GringoCroco, Haakon, Hede2000, Igoldste, Immibis, Infrogmation, JCWilson, Jafeluv, James smith2, Jgiam, Jnc, Johnuniq, Jorge Stolfi, Josh Parris, Karih, Kbrose, Lankiveil, Looxix, MajorFreakinEnglish, Mange01, Mark7-2, MartinHarper, Mhby87, Muhammed Salman Jamal, Muhandes, NYMets2000, Nathan Hamblen, Niteowlneils, Nixdorf, Oscarthecat, Pepsi Lite, QuadrivialMind, RazorICE, Rick Sidwell, RobertMfromLI, SatyrTN, ScottDavis, Strake, Template namespace initialisation script, Tompagenet, Widefox, Woohookitty, Yacht, Zac439, 128 anonymous edits

Out-of-order delivery *Source*: http://en.wikipedia.org/w/index.php?oldid=332547189 *Contributors*: Phatom87, Stwalkerster, The Anome

Packet loss *Source*: http://en.wikipedia.org/w/index.php?oldid=440686361 *Contributors*: AdmJamrep-NJITWILL, Bevo, Debackerl, Frap, Furrykef, GTBacchus, Gurch, Haqpunk, HatlessAtlas, Huyi, Imcdnzl, JZelos, Jalal0, JonHarder, Kenyon, Kglavin, Lee Carre, Liveste, M347758, Mange01, Mctmike, Muhandes, Nimiew, Rbarreira, SlipperyHippo, Spazure, Storkk, Stuartyeates, Tjbk tjb, Tobias Bergemann, V8rik, 52 anonymous edits

Private network *Source*: http://en.wikipedia.org/w/index.php?oldid=461911367 *Contributors*: 33 Degrees, Aboluay, Ajoergensen, Alansohn, AndrewWTaylor, Antonio Lopez, Apokrif, ArchonMagnus, Avono, Awaterl, Big Bosses, Big66, Blanchardb, Blatkinson, Booyabazooka, Bubba nuts, Burtyb, CamD, Cireshoe, Clarince63, Clarus The Dogcow, Corkbob, Csmiller, DSatz, Dandorid, David Gerard, Dayv, DeadEyeArrow, Discospinster, Dmpase, Dotancohen, Dr Zak, Echuck215, Emurphy42, Epbr123, Evilgohan2, Flyer 13, Freedomlinux, GSK, Gaius Cornelius, Generalmiaow, Hairy Dude, Henning Makholm, IOLJeff, Ignacioerrico, Indefatigable, Intelligentfool, Intersofia, JTN, Jackol, Jamelan, Jasper Deng, JeffyJeffyMan2004, JesseW, Joey Novak, KGV, KJBracey, Karn, Karol508, Kate, Kbrose, Kocio, Kvng, Limbo socrates, Lueckenhoff, MER-C, Markaci, Maximus Rex, Mblumber, Mboverload, MeekMark, Mike Stramba, Mindmatrix, Murdockh, Noahspurrier, NuShrike, Nuno Tavares, Oli Filth, Outlyer, Owendelong, Owenjms, Ozzzo, PM Poon, PavlikPolivka, Penguinfan, PhiLiP, Pilatus, ProgFou, Quantumobserver, Qxz, RainbowOfLight, Razwww, Rcsprinter123, RevRagnarok, Rich Farmbrough, Ricky81682, Rkent vincy, Rorx, SDC, SchuminWeb, Scorpiuss, Shambler0, Sigkill, Skew-t, Soliloquial, Sorenriise, Stephan Leeds, StoneIsle, StuffOfInterest, T23c, Tabletop, Temporaluser, The Anome, TheNewPhobia, Tom harrison, TristinGrind, Tverbeek, Uruiamme, Varuna, Whitepaw, WikiShaddow, Wrs1864, Wyksztalcioch, Xoneca, 177 anonymous edits

Route poisoning *Source*: http://en.wikipedia.org/w/index.php?oldid=464528664 *Contributors*: Althena, CSW1986, Circeus, Doctorfluffy, Lendorien, Lls71, Mindmatrix, MrRadioGuy, RedWolf, Roshjack47, Sfisher, Svick, The Wiki Octopus, Upholder, 20 anonymous edits

Subnetwork *Source*: http://en.wikipedia.org/w/index.php?oldid=463151055 *Contributors*: 0612, A520, Aapo Laitinen, Abdull, Aerion, Aicer, Ajo Mama, Alansohn, Alexius08, AlisonW, AlistairMcMillan, Allstone, Andrewpmk, Angr, Anna Lincoln, Aos101, Areguera, AubreyEllenShomo, Aurigas, Azrael81, Babbage, Basil.bourque, Bezenek, Bigmantonyd, Blehfu, Bob.v.R, Boing! said Zebedee, Borgx, Bovineone, Byap, C J B Scholten, Cameltrader, Canderra, Capricorn42, ChiragPatnaik, Christopher P, Connormah, Courcelles, Crimson30, Crossmr, DARTH SIDIOUS 2, DancingPenguin, Danielro, Dart evader, David H Braun (1964), Ddxc, Denisarona, DerHexer, Dgtsyb, DhananSekhar, Djinn112, Doctorfluffy, Domeng, DrSeehas, Dreadstar, Eep², Electrified Fooling Machine, Esrob, Floydpink, Gamera2, Getsnoopy, Glane23, Gothmog.es, Graciella, Grampajoe, Grandmaster e, Greyowl7, Hairy Dude, Halloko, Hcberkowitz, Heracles31, Hoziron, Hu12, In fact, Isidore, Isnow, Izzyp, J.delanoy, JSandbrook, JTN, Jake Nelson, Jdlambert, Jeepday, Jengelh, Jnc, Johnteslade, Johnuniq, JonHarder, Josh Parris, JovBlackheart, Jsg24, Jz32300, Kajaro, Kbdank71, Kbrose, Kdg81, Kirkpthompson, Kku, Klaver, Krellis, Kvng, Kwertii, Kyrikan, L Kensington, Lapsus Linguae, Logictheo, Lovtolaph, Mage Whopper, Mancini, Mandarax, Mange01, Manway, Marc Kupper, Marchash, Mark1000, Methossant, MichaelBillington, Mike.lifeguard, Mikeblas, Mindmatrix, MithrandirAgain, MoodFreak, Mulad, Natkeeran, NawlinWiki, Neilc, NigelJohnson, Nikul padhya, Notmat, Nubiatech, Ohnoitsjamie, OlivierWeb, Omicronpersei8, Outlyer, P0lyglut, PCock, PFHLai, PamD, Pap3rinik, Paul, PedroPVZ, Pennyroyaltea, Phantom784, Philip Trueman, Pontillo, Promethean, Reinderien, Remember the dot, Rick Sidwell, Rimshot, Rjwilmsi, Rob Cranfill, Romeu, Ryucloud, Samdutton, Samwb123, Search4Lancer, Seraphimblade, SidP, Sidious1701, SirGrant, Slant, Snoops, Sonnenhohl, Srbanator, Stannered, Stewartadcock, Stv, T23c, Taboo Tongue, Thankgodimacountrygeek, The Anome, The Bushranger, The ball, TheDoober, Thisisfutile, Thoglette, Tjpayne, Tobias Bergemann, Tommy2010, Tonyhallmailbox, Train2hlp, Tylclement, Unixer, Vapetino, Varlaam, Versus22, Vipinsh001, Vlhsrp, WatchAndObserve, Wayne Slam, Weirdstapler, Weylinp, Wiki alf, Wolftengu, Woohookitty, Wrs1864, Xerxes72, Yonatan, Yrithinnd, Yubal, Zac439, Zero sharp, ZimZalaBim, Zvn, 565 anonymous edits

Unicast *Source*: http://en.wikipedia.org/w/index.php?oldid=453925758 *Contributors*: Amaraiel, Avs5221, Denisarona, Dpaulino, Dr.queso, Easyas12c, Facorread, Hafrul, Interiot, Iohannes Animosus, JForget, JTN, Johnuniq, JonHarder, Joris, Joy, Kai, Karl Lee, Kauczuk, Kbrose, Kortaggio, Kvng, M4gnum0n, Mindmatrix, Neilc, Perhelion, QTCaptain, Quest for Truth, Solde9, Stevech63, Storkk, Suresh creation, Sverdrup, Taichi, Tedernst, Wihwang, 35 anonymous edits

Additive increase/multiplicative decrease *Source*: http://en.wikipedia.org/w/index.php?oldid=461499180 *Contributors*: Bearcat, Bkkbrad, Brucezhezhang, Douggie1085, DustFormsWords, Gumballi, Indil, Kvng, Malcolma, Mdd4696, Mgream, Mvamsip, Plrk, Rivanvx, Rusack, Sophysduckling, Unixxx, Uogl, 17 anonymous edits

Congestion window *Source*: http://en.wikipedia.org/w/index.php?oldid=451010303 *Contributors*: Andre.holzner, C0N6R355, Dinomite, DustFormsWords, Hoverbird, Imcdnzl, Jones Jughead, Justin HW, Ketiltrout, Meghna24, PolarYukon, SunCreator, Underpants, Verdy p, Vortura, Widefox, Zanoni, 25 anonymous edits

Connection-oriented protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=457658803 *Contributors*: Altzinn, Amalas, BAxelrod, Bongwarrior, Epachamo, Jim.henderson, JonHarder, Jpbowen, Jradix, KagamiNoMiko, Kvng, Leszek Jańczuk, Mange01, Mild Bill Hiccup, Mojodaddy, Mpeylo, Ndcuong1980, Paraside, Phatom87, Reedy, W33v1l, 19 anonymous edits

Connectionless protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=461094913 *Contributors*: Abdull, Amalas, BAxelrod, Cburnett, Claude.fachkha, CoupleKlonopin, EagleOne, Epachamo, Fabiovh, JonHarder, Jpbowen, Kgfleischmann, Kito85, Kvng, Madman91, Martin Hinks, Mdsam2, Mojodaddy, Mpeylo, Radagast83, Rajanand online, Reedy, The real dan, 20 anonymous edits

Fast retransmit *Source*: http://en.wikipedia.org/w/index.php?oldid=445012917 *Contributors*: Dyslexic agnostic, Katharineamy, Malcolma, Nealc, P0per, Rahamatullah, SPiNoZA, Suruena, TediousFellow, Zeerak88, 10 anonymous edits

Karn's Algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=362780568 *Contributors*: Adamianash, Andreas Kaufmann, Douggie1085, Jll, Martijn Hoekstra, Nealcardwell, Ngriffeth, 6 anonymous edits

Maximum segment size *Source*: http://en.wikipedia.org/w/index.php?oldid=449108037 *Contributors*: Adoniscik, Adrignola, AlistairMcMillan, Boscobiscotti, Charles Matthews, Claude.fachkha, DagErlingSmørgrav, Deville, Douggie1085, Fresheneesz, Imcdnzl, JonHarder, Julesd, K001, Kermesbeere, Kevin Saff, Khirbat, MichaelGoldshteyn, N328KF, NortyNort, RJFJR, Rich Farmbrough, Ricky81682, Rivanvx, Scetoaux, Sparky132, Suruena, Tsunanet, Waelder, Widefox, Xaje, 20 anonymous edits

Nagle's algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=448036091 *Contributors*: (:Julien:), Andreas Kaufmann, Charles Matthews, Chris Chittleborough, Cmdrjameson, Cwolfsheep, Dom316, Ej, Emesee, Ilmari Karonen, J.delanoy, Jayjg, Jdstroy, Josephholsten, Julesd, Karada, Ketiltrout, Lee Carre, Lowellian, Luc4, Mcnoze, Michaelmior, MoreNet, Nealcardwell, Ordinary Person, Palfrey, Perfgeek, Peterhoneyman, Philipwhiuk, Pkierski, Prezbo, Rjwilmsi, Rogerdpack, RossPatterson, Rror, Sae1962, Sandstein, Scientus, Stephan Leeds, Talldean, Tanketz, Topbanana, Tv0r0g, Woohookitty, Xasodfuih, 55 anonymous edits

Negative-acknowledge character *Source*: http://en.wikipedia.org/w/index.php?oldid=450896341 *Contributors*: Abdull, Againme, Anthony Appleyard, BD2412, Badanedwa, Bevo74, Eastlaw, EdH, Edward, Eupedia, Fordsfords, Fredil Yupigo, Gaius Cornelius, Guy Harris, Hairy Dude, Holi Teacakes, Hooperbloob, IPAddressConflict, Jamsignal, Jim.henderson, Jnc, Kalathalan, Kevin Rector, Mackensen, Mav, Maxim Razin, Mitya89, NeilFraser, PierreAbbat, R'n'B, RainbowCrane, Spitzak, Vadmium, Wolfkeeper, 10 anonymous edits

Network congestion *Source*: http://en.wikipedia.org/w/index.php?oldid=464978433 *Contributors*: Billc.cn, Bloodshedder, Gqqnb, Imcdnzl, Incompetence, Itusg15q4user, Jim.henderson, Jnc, JonHarder, Jtk, Kvng, LachlanA, Leblondleblond, Loftenter, MER-C, Mandarax, Mleoking, Neilc, Nick Number, Nimiew, PabloCastellano, QmunkE, Rememberway, Seifried, Stw, The Anome, The Evil IP address, Woohookitty, Wrs1864, Xofc, Тиверополник, 19 anonymous edits

Path MTU Discovery *Source*: http://en.wikipedia.org/w/index.php?oldid=445473426 *Contributors*: CesarB, Dicentra, Dr. Blofeld, JonHarder, Karlhiramoto, Kbrose, Kenyon, Kvng, Machilde, Martarius, Oayfer, The Anome, Tobias Bergemann, Wrs1864, Xmm0, 20 anonymous edits

Port number *Source*: http://en.wikipedia.org/w/index.php?oldid=463638600 *Contributors*: 48v, AdjustShift, Allen4names, Arbitrary username, Arcturus4669, Artyom, Bobo192, Booyabazooka, Borgx, Brian Crawford, BrianWren, Cambrant, Chester Markel, Chris24359, ChrisHodgesUK, Coolcaesar, Daniel Staal, Dcljr, Dicklyon, Dmit, Edward, Elronxenu, Etherknot, Fastily, Flewis, Flow3r, FrankTobia, Fufthmin, Furrykef, Grammar conquistador, Helix84, Helvetica, Incnis Mrsi, Int21h, Intelligentsium, Jfmantis, Jncraton, John D Burling, Jonathanvt, Kbrose, Kevin82485, Kvng, Lampak, Lee Carre, Linnando, Maheshcms, MattGiuca, NonvocalScream, Northumbrian, NotAnonymous0, Nurg, Orphan Wiki, Oxymoron83, PanagosTheOther, Pde, Phatom87, Philip Trueman, Phuzion, Pinethicket, Pred, Pxma, Radagast83, Redsxfenway, Riki, RobertL, Sanguinia, Seraphiel, Shanes, Sparsh Nagpuri, Spur, Srain, TMC1221, Teddks, Teryx, TheParanoidOne, Tom Y8s, Tslocum, Tswsl1989, Uruiamme, Vanni.T, Versus22, Vigilius, Whitehatnetizen, Zack3rdbb, 178 anonymous edits

Random early detection *Source*: http://en.wikipedia.org/w/index.php?oldid=462368833 *Contributors*: AdrianSuter, Alvin-cs, Ashok.A-K, Cmw1, Dcoetzee, Glenn, Helix84, Henrygb, Jamelan, JayCarlson, Jec, Joseanes, Joy77, Julesd, Kvng, Kwid, Lear's Fool, Leastfixedpoint, Markaci, Michael Hardy, Mleoking, Msuzen, Pratick.chokhani, R6144, Rangeenbasu, Rjwilmsi, Runtime, Spearhead, Spmeyn, Tikiwont, Vrable, Woohookitty, Xasodfuih, ZeroOne, 42 anonymous edits

Retransmission (data networks) *Source*: http://en.wikipedia.org/w/index.php?oldid=461982456 *Contributors*: Avalon, Discoveralaska, Douggie1085, Guy Harris, HiB2Bornot2B, Ixfd64, Jnc, Mange01, Obakeneko, Ryan Roos, Soumyasch, Storkk, Wrs1864, 19 anonymous edits

Round-trip delay time *Source*: http://en.wikipedia.org/w/index.php?oldid=454469659 *Contributors*: Ahassan503, C xong, Danogo, Daviey, Dawynn, Deville, Dismas, Dto, Felix3, Floul1, GPHemsley, Guy M, Harry, Heron, Hooperbloob, Ilya78, Iridescent, Jim.henderson, Kku, Ndemou, Ngriffeth, Patrick, Pdcook, Rcronk, Remotelysensed, Rivanvx, Rursus, Salvarez83, Shengzhongxie, Tobias Bergemann, Tony Sidaway, TreveX, Wernher, Ztepsic, İbrahimbarbaros, 41 anonymous edits

Silly window syndrome *Source*: http://en.wikipedia.org/w/index.php?oldid=462841760 *Contributors*: Acdx, Andrew Hampe, Apoc2400, Arthena, Bisqwit, Cckid, Chris83, Dan2119, Douggie1085, EoGuy, Greenrd, Icycle, Illythr, Jll, Kunmor, Mark Renier, Nneonneo, Orange Suede Sofa, Orborde, PDH, Perchy22, Praveraj, Scott McNay, Splintercellguy, The Anome, Tizio, Warren, Widefox, Wtmitchell, 13 anonymous edits

Sliding window protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=453228623 *Contributors*: Andy Dingley, Arthena, Bobnorwal, Chris the speller, Cimp3, DarkPhoenix, Difu Wu, Fightin' Phillie, Fratrep, Georgeg2481, Ibc111, Ilmari Karonen, Jacobko, Jheiv, Johnuniq, Kbrose, LittleBenW, MER-C, Nimiew, Oli Filth, Phatom87, Ryanl, Samitshah1, Shisock, Siddhant, SpikeToronto, Storkk, Widefox, Zoonfafer, Тиверополник, 73 anonymous edits

Slow-start *Source*: http://en.wikipedia.org/w/index.php?oldid=425953655 *Contributors*: Asteiner, Bachrach44, DonDiego, Fresheneesz, Fubar Obfusco, Icecreamguy, Imcdnzl, Josh Parris, Ketiltrout, Klemen Kocjancic, LOL, Longerro, MER-C, Marchash, Markhurd, Mdd4696, Merinith, Moritz, Neilc, Nuno Tavares, P0per, Pietrodn, PolarYukon, R. S. Shaw, Rb82, Ricky Lau, Slaniel, ThomasBosboom, Tobycat, Whitepaw, Zwaen, 57 anonymous edits

TCP congestion avoidance algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=464223120 *Contributors*: Ahassan503, Anthony Rushton, Barvinok, Beren023, Bfitz99, Bhy, Blaisorblade, Bwinds, Cawka, CesarB, Cheskis, Coinchon, Cxxl, Doristein, Dweekly, Dysprosia, Foobaz, Frap, Grievous Angel, Herr Baum, Imcdnzl, Intgr, Jedonnelley, Jibjibjib, Jimreynold2nd, Jugander, Ketiltrout, Kvng, LachlanA, Lavendert, Madhupsinha, Mess, Mgream, Michael Hardy, Michael miceli, Mpeylo, Musiphil, Nealcardwell, Nimiew, NovaScorpio, Oldiowl, Plaidomatic, Rememberway, Ricky Lau, Robosycho, RyanCu, Sam Hocevar, Shaddack, Sligocki, Soumyasch, SystemBuilder, The Anome, TheParanoidOne, Tmimak, Toothswung, Treekids, Uogl, Whitepaw, Winterheart, Xcyborg, YUL89YYZ, 102 anonymous edits

Transmission Control Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=463245067 *Contributors*: 10metreh, 5 albert square, A-moll9, A1vast, A5b, Access Denied, Acdx, AgadaUrbanit, Agent Koopa, Agi896, Ahson7, Akamad, Akill, Akshaymathur156, Alca Isilon, Aldie, Ale2006, Alex, Alexescalona, Alexh19740110, Alexius08, Alt-sysrq, Alvin-cs, Amalthea, Andre Engels, Andy M. Wang, Anichandran.mca, Anna Lincoln, AnotherNitPicker, Anwar saadat, Aprice457, Arnavchaudhary, Arnhemcr, Arsenal9boi, Asenine, Ashwinsbhat, Astronomerren, Avono, Awy997, B4hand, BAxelrod, Beccus, Beland, Betterworld, Bezenek, Bigdumbdinosaur, Bilbo1507, Bill Malloy, BillMcGonigle, Biot, Bkell, Bkkbrad, Blacksqr, Blanchardb, Boothy443, Boscobiscotti, Brech, Brighterorange, Brion VIBBER, Bstrand, Bubba hotep, Butros, Can't sleep, clown will eat me, CanadianLinuxUser, Canthusus, Cbrettin, Cburnett, Centrew, Charles.partellow, Chealer, Cheburashka, Christopher P, Cincaipatrin, Ciprian Dorin Craciun, Cjdaniel, Clark-gr, CobbSalad, Coinchon, Colonies Chris, Cometstyles, Conversion script, Cpaasch, CraSH, CrizCraig, D235, DKEdwards, Daev, Daniel Staal, Daniel.Cardenas, Danielbarnabas, Danielgrad, DariuszT, DarkAudit, DaveSymonds, David.bar, Dcoetzee, DeadEyeArrow, Dewet, Dgreen34, Dharmabum420, Dina, Djdawso, Dnas, DnetSvg, Doc Strange, Dondegroovily, Dori, DrHannibal216, Drake Redcrest, Duckbill, DylanW, Egg, Eirik (usurped), El pak, EncMstr, Encognito, Enjoi4586, Enviroboy, Epbr123, Equendil, Eric-Wester, Erik Sandberg, Evice, Evil Monkey, Explicit, FGont, Fabiob, Fawcett5, Fcp999, Fishal, Fisherisland14, Fleminra, Foobaz, Fred Bradstadt, Fredrik, Fredrikh, Frencheigh, Fresheneesz, Fsiler, Fubar Obfusco, Gaius Cornelius, GalaxiaGuy, Gamera2, Garion96, GarryAnderson, Ghettoblaster, Giftlite, GilHamilton, Glenn Willen, Gmaxwell, Goplat, Graeme Bartlett, Graham87, Guy Harris, Gwinnadain, Haggis, Hairy Dude, Hamtechperson, Harput, HarrisonLi, Harryzilber, Harvester, Helix84, Henriktudborg, HiB2Bornot2B, I already forgot, I-baLL, IOLJeff, IRP, Idcmp, Ideoplex, Iitywybmad, Imcdnzl, Imroy, Inter, Intgr, J. Nguyen, JCO312, JTN, Jaan513, Jackol, Jec, JeffClarkis, Jehorn, Jewbacca, Jgeer, Jgrahn, Jjinno, Jnc, JoanneB, Jogers, John Vandenberg, Johnuniq, JonHarder, Jonathan Hall, Jondel, Jowagner, Jsavage, Jtk, Juliancolton, JuneGloom07, Jusdafax, Jxw13, Karada, Kartano, Kasperl, Katimawan2005, Kbrose, Kenyon, Kevinmon, Kgfleischmann, Kim Bruning, Kinema, Klhuillier, Krellis, Kubanczyk, Kumarat9pm, Kvng, Kwamikagami, Kwiki, L Kensington, LachlanA, Lam Kin Keung, Lanilsson, Lark ascending, LeiZhu, LeoNomis, Leolaursen, Leszek Jańczuk, Leyo, LiDaobing, Lightdarkness, Lights, LilHelpa, Lilac Soul, Lime, Logicwax, Lokacit443, Longuniongirl, Loor39, Loukris, Ltsampros2, Lucasbfr2, Luk, Luna Santin, Lunadesign, Lupo, M.S.K., MER-C, Mac, MacStep, Maclion, Maerk, Maimai009, Makomk, Maksym.Yehorov, Mange01, Manlyjacques, Manop, Marek69, Mark Bergsma, Markrod, Martijn Hoekstra, Marty Pauley, Materialscientist, Mattabat, Mboverload, Mbruck, MeekMark, Meekohi, Mendel, Mhandley, Michael Frind, Michael Hardy, Mild Bill Hiccup, Miss Saff, MithrasPriest, Mjb, Mohitjoshi999, Mootros, MoreNet, Mozzerati, Mr Echo, MrBoo, MrOllie, Mrzehak, MtB, N328KF, NKarstens, Nateshwar kamlesh, Nave.notnilc, NawlinWiki, Nealcardwell, Nedim.sh, NeilGoneWiki, Neo139, Neshom, Nestea Zen, Ngriffeth, Nifky?, Nikhil raskar, Nikola Smolenski, Nimiew, Nixdorf, Nk, Nneuman, Northamerica1000, Nubiatech, Ocaasi, Ojs, Okiefromokla (old), Oldiowl, Omicronpersei8, Ordoon, Ortzinator, Otets, Ouimetch, Oxyacanthous, Oxymoron83, P0per, PBSurf, PS3ninja, Pak21, Palica, PanagosTheOther, Papadopa, PaulWay, Pde, PedroPVZ, Pegasus1138, Perfgeek, PeterB, Peytons, Pfalstad, Pg8p, Pgr94, Phandel, Phantomsteve, Phatom87, PhilKnight, Phuyal, Piano non troppo, Plainsong, Plugwash, Pluknet, Pmadrid, Poeloq, Prakash mit, Prasannaxd, Prashant.khodade, Prunk, PureRumble, Purplefeltangel, Putdust, PxT, Quantumobserver, Qutezuce, Qz, RA0808, Raul654, Rdone, RedWolf, Rettetast, RevRagnarok, Revolus, Rick Sidwell, Rjwilmsi, Rodowen, RodrigoCruzatti, Rogper, RoyBoy, Rtclawson, Ru.in.au, Ryan Stone, SHIMONSHA, SWAdair, Saaga, SamSim, Samuel, Sasha Callahan, Savagejumpin, Sbnoble, Scientus, Scil100, Scorpiondiain, Scrool,

Sdomin3, Sepper, Sergiodc, Sh manu, Shaddack, Shadowjain, Shadowjams, Shultzc, Sietse Snel, Sim, Sleigh, Smappy, Smyth, Snowolf, Some Wiki Editor, Spearhead, SpeedyGonsales, Splint9, Spoon!, Squidish, StephenHemminger, Stephenb, Stevenwagner, Stonesand, StradivariusTV, StubbyT, Suruena, Svinodh, Synchrite, Syp, THEN WHO WAS PHONE?, TangentCube, Tariqabjotu, Tasc, Technobadger, TediousFellow, Teles, Template namespace initialisation script, Tfl, The Anome, The Monster, The Thing That Should Not Be, The-tenth-zdog, TheVoid, Tide rolls, Timotheus Canens, Timwi, Toh, Tomchiukc, Tommy2010, Torla42, Trou, TuukkaH, Umar420e, UncleBubba, Unixguy, Urmajest, Ursushoribilis, Uruiamme, Vedantm, Via strass, Vinu Padmanabhan, Vrenator, WLU, WikHead, WikiLaurent, Wimt, Wkcheang, Wlgrin, WojPob, Wolfkeeper, Woohookitty, Wrs1864, Xaphnir, Yonatan, Youpilot, Yyy, Zachlipton, ZeroOne, Zeroboo, Zundark, Zvar, شات صوتي, 1082 anonymous edits

Transport layer *Source*: http://en.wikipedia.org/w/index.php?oldid=458761082 *Contributors*: 1ForTheMoney, Adoniscik, Alansohn, Aldie, AlistairMcMillan, Altenmann, Alvestrand, Andrew Hampe, Andreyf, BenBreen2003, Bigmantonyd, Borgx, Brettmeyers, Butko, Butlerm, Choudesh, Conan, Costello, Dgreen34, Dgtsyb, Eggnock, EnOreg, Enjoi4586, FatalError, Fmunshi, Fred Bradstadt, Gavinatkinson, GermanX, Giftlite, Guy Harris, Hadal, Hede2000, Highlandsun, Hullbr3ach, Imcdnzl, Int21h, Ipatrol, J.delanoy, JHolman, Jafeluv, Jec, Jimfbleak, Jnc, Johnuniq, Kadin2048, Kbolino, Kbrose, Kingpin13, Kkkdc, Kungfuadam, Kvng, Kwi, LiDaobing, Limbo socrates, Looxix, Lost.goblin, MTC, Mange01, Marcan, MartinHarper, Mirv, Mmmready, Nealmcb, Nixdorf, Ovy, Phantomsteve, Phatom87, PrologFan, Razorflame, Retroguy90, Rmhermen, Robth, Samjoopin, SatyrTN, ScottDavis, Simon04, Suffusion of Yellow, Suruena, Tcmcfaul, Template namespace initialisation script, Tpvibes, Vipinhari, Wmasterj, Yacht, Yyy, Zac439, ZeroOne, Zoicon5, 139 anonymous edits

User Datagram Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=465427865 *Contributors*: 28421u2232nfenfcenc, A5b, Aapo Laitinen, Alfio, Ali@gwc.org.uk, Alison22, Alvin-cs, Aram33, Ardonik, Ashmoo, Ayeroxor, BAxelrod, Beacon11, Beefman, BenAveling, Bender235, Benqmonitor, Bentogoa, Bobby Caudwell, Bobstopper, Borgx, Bryan Derksen, Burisch, Cassan, Cburnett, Chenzw, Chiefmanzzz, Cincaipatrin, Conrad.Irwin, Conversion script, CraSH, CrookedAsterisk, Cvalda, DSatz, Damian Yerrick, Daniel Staal, DavidSol, DeadEyeArrow, DennyColt, DesTroy, Dfurbeck, Dicentra, Djordjes, Dkogh, E Wing, E090, Eagleal, Eclipsed, Enjoi4586, Erik Garrison, Esmond.pitt, Falcon9x5, Fennec, Fightin' Phillie, Flewis, Fredrikh, FrisoB, Frodet, Fromageestciel, Fubar Obfusco, Fudgefr, Gamera2, Giftlite, Ginkgo100, Gnipahellir, Golwengaud, Gordoni, Gorryf, Gracefool, Graham87, Gsmgm, Guy Harris, Hdante, Headbangerkenny, Hitherebrian, I dream of horses, Icestorm815, Ideoplex, InverseHypercube, Irrypride, Itai, JNighthawk, JTN, Jengelh, Jjinno, Jncraton, Joachim Wuttke, Joanjoc, Johnny 0, Johnuniq, Jtk, Jóna Þórunn, Kbrose, Kgfleischmann, Kinema, Kovianyo, Krellis, Kvng, Kwamikagami, Kyng, Lissajous, Locos epraix, MLeb, Mahjongg, Mam711, Mange01, Mani1, Materialscientist, Matt Darby, Michael Zimmermann, Mike Dill, Mike6271, Misterdom, Modulatum, Mpeg4codec, Mwholt, Neilc, Nixdorf, Nroets, Nubiatech, Od Mishehu, Ouimetch, Oxymoron83, Paulish, PedroPVZ, Pelesl, Petri Krohn, Phatom87, Plugwash, Public Menace, Quiddity, R2richar, Razorflame, Rebecca, Recurring dreams, RedWolf, Rememberway, Reub2000, RevRagnarok, Reywas92, Rick Sidwell, Robbe, RobertL30, Romanm, Ronz, Rwwww, S-n-ushakov, SWAdair, Samersarhan83, Sammydee, Samuel, Sci13960, Smeggysmeg, Spearsall, Ssd293, Suruena, Tdangkhoa, Teemuk, Teh roflmaoer, Template namespace initialisation script, The Anome, The Monster, The Utahraptor, The-tenth-zdog, Thray, Tjdw, Tobias Bergemann, Towel401, Tpbradbury, Twinxor, UncleBubba, Upadhyaykc, Vinu Padmanabhan, Voidxor, WestwoodMatt, Wiarthurhu, Wkcheang, Xezbeth, Zara1709, ^demon, Александър, Тивеополник, 328 anonymous edits

Adaptive routing *Source*: http://en.wikipedia.org/w/index.php?oldid=433720724 *Contributors*: CesarB, Eriberto mota, Itai, Jnc, Kamalsinghpanwar, Materialscientist, Pgallert, Think outside the box, Timo Honkasalo, 19 anonymous edits

Bellman–Ford algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=461552478 *Contributors*: Aaron Rotenberg, Abednigo, Aene, Agthorr, AlexCovarrubias, Altenmann, Anabus, Andris, Arlekean, B3virq3b, Backslash Forwardslash, BenFrantzDale, Bjozen, Bkell, BlankVerse, Brona, CBM, Charles Matthews, CiaPan, Ciphergoth, David Eppstein, Dcoetzee, Docu, Drdevil44, Ecb29, Enochlau, Epbr123, Ferengi, FrankTobia, Fredrik, Fvw, Gadfium, Giftlite, GregorB, Guahnala, Headbomb, Heineman, Helix84, Iceblock, Istanton, Itai, J.delanoy, Jamelan, Jaredwf, Jellyworld, Jftuga, Josteinaj, Justin W Smith, Konstable, LOL, Lavv17, Mario777Zelda, Mathmike, Mazin07, Michael Hardy, Miym, N Shar, Naroza, Nihiltres, Nils Grimsmo, Orbst, P b1999, PanLevan, Pion, Pjrm, Poor Yorick, Posix4e, Pskjs, Quuxplusone, Rjwilmsi, RobinK, Rspeer, SQL, Salix alba, Sam Hocevar, Sigkill, Solon.KR, SpyMagician, Stdazi, Stderr.dk, Stern, Str82no1, Tomo, ToneDaBass, Tsunanet, Tvidas, Waldir, Wmahan, Writer130, Zholdas, 147 anonymous edits

Border Gateway Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=465192119 *Contributors*: 2D, AS, Adamantios, Ahoerstemeier, Aldie, Alexius08, Andareed, Anna Lincoln, Aphstein, Ari.takanen, Aryaniae, Ash71p, Astronouth7303, Balabiot, Barrylb, Benjamin Pineau, Bergonz, Betacommand, Bfigura's puppy, Bill.albing, Bobo192, Breno, CBM, Can't sleep, clown will eat me, Charm, Chealer, Chochopk, Chris the speller, Cluth, Colfer2, Crazycomputers, DESiegel, DanielSiva, Dannofarrell, DarkCow, David Eppstein, DavidDouthitt, Daydr3am3r, Dgtsyb, Dguye, Digitalme, Dimawik, Dinomite, Dondegroovily, EagleEarth, Edward, ElBenevolente, Elder, Emccrckn, Equendil, Erikrespo, Fontas, Foobaz, Frazzydee, Fsamuels, Fsiler, Func, Furby100, Gail, GastonSenac, Geeoharee, Georgesawyer, Gfrank, Giftlite, Gnebulon, Gonchibolso12, Hads1, Hairy Dude, Hashar, Hcberkowitz, Htaccess, Hughcharlesparker, Humble226, IMSoP, Iamregin, Ianeiloart, Igxx, IncognitoErgoSum, Inkling, JHunterJ, JTN, Jaeger5432, JamesBWatson, Jandalhandler, Janizary, Japanese Searobin, Jarsyl, Jbond00747, Jgs42, Jnc, Joeinwap, Joel7687, JohnOwens, Johnthomastaylor, Jon Awbrey, Jwvo, Kbrose, Kfor, Kinema, Kll, Klykken, Korstellan, Krellis, Kvng, Kyleflaherty, L Kensington, LaMenta3, Laug, Leebert, Lentoyip, Lessaid, Lightmouse, Llywrch, Luder, Lukeritchie, MDW19873, Marcod'Itri, Marokwitz, MassimoAr, MathsPoetry, MatthewWilder, MaverickSolutions, Mblumber, Michael Hardy, Michael miceli, Mike Rosoft, Miym, Mozzerati, Mro, Muhandes, Nealmcb, Negrulio, Nethgirb, Nightstallion, Nihiltres, Nikai, Nneonneo, Nubiatech, Nv8200p, Oxymoron83, Petri Krohn, Phatom87, Philip Trueman, Phillipgrasso, Prolixium, Qr.brown, Rdummarf, Redquark, Reedy, RekishiEJ, Requestion, Rhondle, Ric8ard, Rich Farmbrough, Robert Brockway, Ronk01, Root2, Ruud Koot, Schneelocke, Sepiraph, Sherocksfriend, Sigkill, Signalhead, Sim, Snarius, So14k, Southpaw018, Sspecter, Strait, Super Jamie, Suruena, Tekhammer, Tekhnofiend, Telecomtom, Termininja, The Anome, The Thing That Should Not Be, Thedangerouskitchen, Thiseye, Thruston, Tjmao, Tjpayne, Tm knoll, Tmaufer, TreyGeek, Ttwaring, Vary, Vcappuccio, Versus22, Via strass, Vjardin, Wavelength, Woohookitty, Wrs1864, Xipher, Xoxon1kk1, Yllt, Zakawer, Zalnas, ZippyCycle, 395 anonymous edits

Dijkstra's algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=465308439 *Contributors*: 4v4l0n42, AJim, AK Auto, Abu adam, Adamarnesen, Adamianash, AgadaUrbanit, Agthorr, Ahy1, AlanUS, Alanb, AlcoholVat, Alex.mccarthy, Allan speck, Altenmann, Andreasneumann, Angus Lepper, Anog, Apalamarchuk, Aragorn2, Arrenlex, Arsstyleh, AxelBoldt, Aydee, B3virq3b, B6s, BACbKA, B^4, Bcnof, Beatle Fab Four, Behco, BenFrantzDale, Bgwhite, Bkell, Blueshifting, Boemanneke, Borgx, Brona, CGamesPlay, CambridgeBayWeather, Charles Matthews, Christopher Parham, Cicconetti, Cincoutprabu, Clementi, Coralmizu, Crefrog, Css, Csurguine, Cyde, Danmaz74, Daveagp, David Eppstein, Davub, Dcoetzee, Decrypt3, Deflective, Diego UFCG, Digwuren, Dionyziz, Dmforcier, Dmitri666, Dosman, Dreske, Drostie, Dudzcom, Dysprosia, Edemaine, ElonNarai, Erel Segal, Eric Burnett, Esrogs, Ewedistrict, Ezrakilty, Ezubaric, Faure.thomas, Fox, FrankTobia, Frankrod44, Fresheneesz, GRuban, Gaiacarra, Galoubet, Gerel, Geron82, Gerrit, Giftlite, Gordonnovak, GraemeL, Graham87, Grantstevens, GregorB, Guanaco, Gutza, Hadal, Haham hanuka, Hao2lian, Hell112342, HereToHelp, Herry12, Huazheng, Ibmua, Illnab1024, Iridescent, Itai, JBocco, JForget, Jacobolus, Jaredwf, Jason.Rafe.Miller, Jellyworld, Jeltz, Jewillco, Jheiv, Joelimlimit, Jongman.koo, Joniscool98, Jorvis, Julesd, Justin W Smith, K3rb, Kbkorb, Kesla, Kh naba, Kku, Kndiaye, Kooo, Kostmo, LC, LOL, Laurinkus, Lavaka, Leonard G., LordArtemis, LunaticFringe, Mahanga, Mameisam, MarkSweep, Martynas Patasius, MathMartin, Mathiastck, MattGiuca, Matusz, Mccraig, Mcculley, MementoVivere, Merlion444, Mgreenbe, Michael Slone, Mikeo, Milcke, Mkw813, MusicScience, Mwarren us, Nanobear, NetRolller 3D, Nethgirb, Nixdorf, Noogz, Obradovic Goran, Obscurans, Oleg Alexandrov, Oliphaunt, Optikos, Owen, PesoSwe, Piojo, Pol098, PoliticalJunkie, Possum, ProBoj!, Pseudomonas, Pshanka, Pskjs, Pxtreme75, Quidquam, RISHARTHA, Radim Baca, Ramajois, Rami R, RamiWissa, Recognizance, Reliableforever, Rhanekom, Rjwilmsi, Robert Southworth, RodrigoCamargo, RoyBoy, Ruud Koot, Ryan Roos, SQGibbon, Sambayless, Sarkar112, Shd, Sheepeatgrass, Shizny, Shuroo, Sidonath, SiobhanHansa, Sk2613, Slakr, Sniedo, Sokari, Someone else, Soytuny, Sprhodes, Stdazi, SteveJothen, Styath, Subh83, Sundar, Svick, T0ljan, Tesse, The Arbiter, TheRingess, Thijswijs, ThomasGHenry, Timwi, Tobei, Tomisti, Torla42, Turketwh, VTBassMatt, Vecrumba, Vevek, Watcher, Wierdy1024, WikiSlasher, Wikipelli, Wildcat dunny, Wphamilton, X7q, Ycl6, Yworo, ZeroOne, Zhaladshar, Zr2d2, 538 anonymous edits

Distance-vector routing protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=463812683 *Contributors*: Amux, Atypic, Bcohea, Bobo192, Brighterorange, Christophenstein, Climber22, Cwwang, Cícero, Demitsu, Dennypayne, Derbeth, DonDiego, Ed g2s, Elonka, Fgcscm, Firezx, Gamera2, Gigabite2, Hadal, Helix84, Infinitesteps, Iridescent, Itai, Ixfd64, Jab843, JamesBWatson, Jec, Jnc, Kenyon, Kvng, L6yang, Lembut, Less Than Free, Mhnin0, Nbachiyski, Nbeckman, Neilc, Nethgirb, Netravathi, Nixdorf, Odoncaoa, Palfrey, Pecunia, Phosphoricx, Piet Delport, Pipelynx, RG2, RPHv, Rhyvun, Rita.dos.santos, Rory096, Scraimer, The Anome, Tommy2010, Tonyboy, UMwoodr0, VMS Mosaic, Vanished user 39948282, Veysiertekin, WPjcm, Wolfkeeper, Wonglijie, 169 anonymous edits

Enhanced Interior Gateway Routing Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464890967 *Contributors*: Aeluwas, Aldie, Amore proprio, Andrei Stroe, Belanidia, Bensykesuk, Boivie, Caruch6392, Cit helper, Cpetty-wiki, Cwolfsheep, Daedalus01, DanielSHood, Dejaphoenix, Dgtsyb, Diamond2, Dinomite, Dogcow, Ekeb, Gurch, Iamregin, Icemaniceman1111, Jackfork, Jaesonv, Jamelan, Jec, Jesse Viviano, JidGom, Jnc, Kbrose, Klykken, Kvng, Legoktm, Lightmouse, Lipatden, MER-C, Mgeorge27, Mstuomel, Neilc, Niteowlneils, Noytza, Oscarthecat, Paluchpeter, Pgallert, Pgr94, Phatom87, Pit, Psz, PxT, Qrsdogg, RedWolf, Rhaas, Rich Farmbrough, RichiH, Rjwilmsi, Rohank9, RuED, Sadads, Sam Korn, Shadowjams, Shahid789, Steveoelliott, Takanoha, The Thing That Should Not Be, UncleBubba, Unyoyega, Victor--H, Vlospf, Wrc60, Yellowweasel, 134 anonymous edits

Exterior Gateway Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=449134151 *Contributors*: A2Kafir, Bouchecl, DESiegel, Fireman biff, Halfloaded, IceKarma, JTN, Jec, Jon Awbrey, Jonathanchaitow, Joonasl, Kvng, Mercury543210, Nealmcb, Orphan Wiki, Pilotguy, Rainnsurf, Raul654, Redvers, Sfisher, Squall848, Unyoyega, Vary, Weylin.piegorsch, Wikiti, Wk muriithi, Wrs1864, Y.P.Y, Zyqqh, 26 anonymous edits

Flooding (computer networking) *Source*: http://en.wikipedia.org/w/index.php?oldid=440154680 *Contributors*: BradBeattie, Hydrox, Jpeob, Kitplane01, Ktr101, L3prador, Last Avenue, Maura Dailey, Maxime.Debosschere, Michael Slone, Mmernex, Sengkang, Storkk, 9 anonymous edits

Flooding algorithm *Source*: http://en.wikipedia.org/w/index.php?oldid=448734902 *Contributors*: Ankit kulkarni, Anon423, Dragonchilde, Dysprosia, Graham87, Jaredwf, Jeodesic, Jk2q3jrklse, Michael Slone, PhilippWeissenbacher, Piet Delport, Samois98, SuperFLoh, 36 anonymous edits

Hop count *Source*: http://en.wikipedia.org/w/index.php?oldid=459848586 *Contributors*: Bearcat, Damian Yerrick, Deville, Dthomsen8, Eclecticology, Etincelles, Homestarmy, Hooperbloob, Kbrose, Owen, PaulHanson, PhilKnight, Pots1012, Rails, Stagira, Steven jones, 8 anonymous edits

Interior gateway protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=455730830 *Contributors*: ALM scientist, Abrahami, Aldie, Anon lynx, Ash108, Biot, CapitalR, CesarB, Danielbarnabas, Desnacked, Diverman, Fireman biff, Hads1, Itai, Jon Awbrey, Ksn, Kvng, LeeHunter, Markofvero, Materialscientist, Nealmcb, Nethgirb, Nikosdimos, Pchov, Pgallert, Phatom87, Rrreese, Shiryaev, The Thing That Should Not Be, Wk muriithi, 63 anonymous edits

Interior Gateway Routing Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=445045319 *Contributors*: Abrahami, Akaase, Aldie, Aleksandrit, Bcaff05, Bcohea, BenFrantzDale, Birczanin, Conversion script, Costello, Couriant, Cwolfsheep, DARTH SIDIOUS 2, Denisarona, Dgtsyb, Dinomite, Echoray, Fordan, Hilmarz, Htaccess, Iamregin, J veitch uk, Jamelan, JamesBurns, Jec, Jesse Viviano, Jnc, JonHarder, Joy, Katherine, Kbrose, Klykken, KnightRider, Kocio, Kprice10, Metaclassing, Mmclements, Nick Number, Nixdorf, Nuno Tavares, Nux, PeterMottola, Pgallert, Pgr94, Piet Delport, Shahid789, Skunkboy74, SpaceFlight89, Takanoha, Trevor Andersen, Tristanb, Webmaestro, WibblyLeMoende, Woohookitty, Xernous, 90 anonymous edits

IS-IS *Source*: http://en.wikipedia.org/w/index.php?oldid=459396751 *Contributors*: Aldie, Althena, Amore proprio, Baaloo, Bdh4, Biot, Cameronhmatthews, Cwolfsheep, Cycle au, Dgtsyb, Ehn, Facekhan, FleetCommand, GerardM, Hcberkowitz, Hermes11111, Iamregin, Itai, JHunterJ, JTN, JanCeuleers, JebeddiahSpringfield, Jgs42, JidGom, Jtdouglas, Kbrose, LachlanA, Marek69, MarkWahl, MrOllie, Muhandes, Niteowlneils, Paluchpeter, Pfkaplan, RedWolf, Rjwilmsi, Sigkill, Skandha101, Some jerk on the Internet, Tcncv, The Anome, Vjardin, West London Dweller, Wikiacc, Wk muriithi, Woohookitty, Wrs1864, YUL89YYZ, 85 anonymous edits

Link-state advertisement *Source*: http://en.wikipedia.org/w/index.php?oldid=441964061 *Contributors*: Absemindprof, Breakingaway, Disavian, Dlyons493, Ephraim33, Ibc111, Jacky.yang, Johnuniq, JonHarder, Kbrose, Kolyma, Mattisse, MessiFCB, Mild Bill Hiccup, Paluchpeter, Pellythomas, Prolixium, Rinov22, Sptextor, Ttwaring, Wbenton, Westfall, Wk muriithi, Ziguang, 25 anonymous edits

Link-state routing protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=454146604 *Contributors*: ABSOLUTAMENTE, Aleksey Gerasimov, Alvestrand, Bilbo1507, CambridgeBayWeather, Chris Roy, Chris the speller, Chrisjameskirkham, Devaki.vamsi, DonaldEastlake3, Dudesleeper, Dummor, Ejrh, Gamera2, Garosenb, Hoju, Itai, Jackfork, Jaw959, Jnc, Joel.Conover, Jon Awbrey, Karn, Kbrose, Ketil3, Konstable, MC10, Naddy, Nbeckman, Nethgirb, Nixdorf, Omarecd, Pdelong, PhilKnight, Philip Trueman, Pmarti, QEDquid, Random contributor, RedWolf, Roeme, Rōnin, Scytale.de, Spookycave, Tangotango, Thesavagenorwegian, Timo Laine, User2004, Victor--H, Vyzasatya, Wbenton, Woohookitty, 141 anonymous edits

Metrics (networking) *Source*: http://en.wikipedia.org/w/index.php?oldid=461714186 *Contributors*: CosineKitty, Helix84, Iandiver, Jawnsy, Jjurski, Jncraton, JonHarder, Mcoupal, Mikeblas, Ollie, Palthainon, Rich257, SlimRindy, The last username left was taken, Umbertoumm, 16 anonymous edits

Multipath routing *Source*: http://en.wikipedia.org/w/index.php?oldid=422704198 *Contributors*: Brynosaurus, Ihcoyc, KathrynLybarger, PabloCastellano, Poodah, Tangsong, Tinucherian, Ultraexactzz, 2 anonymous edits

Open Shortest Path First *Source*: http://en.wikipedia.org/w/index.php?oldid=464894056 *Contributors*: A Meteorite, ALargeElk, Agreenhalgh, Aldie, Alexf, Alisterb, Andrei Stroe, Andyluciano, Anon lynx, Antonvanderleun, Anuragkothari, Ari.takanen, Astralblue, Attilabedo, Avto.prangulashvili, AxelBoldt, Babinos30, Biot, Blaxthos, Bomazi, Bradleyem, Breakingaway, Brianga, C46, Can't sleep, clown will eat me, Caruch6392, Catamorphism, Celinama, Chealer, Cwolfsheep, Dandorid, Darthsco, Deineka, Dgtsyb, Dicklyon, Dinomite, Dougluce, Dysprosia, EJSawyer, Enjoi4586, Foobarnix, Franboon, Fudoreaper, Giftlite, H8a1r18i9, Hads1, Hangin10, Hashar, Hcberkowitz, Hede2000, Hilmarz, IMSoP, Iamregin, Ibc111, Insanity Incarnate, JHunterJ, JTN, Jac16888, Jakub Horky, Janizary, Javifs, Jawsper, Jeffreyahaines, Jerome, JialiangGao, Jkurynny, Jnc, John, Johnuniq, Jon Awbrey, JonHarder, Joy, JtiToo, Kbrose, Kcoyle, Kenyon, Kinema, Kingpin13, Knowledge512109, Kozuch, Krypticmind, Lightmouse, Lotje, Luc4, MARQUIS111, MarsRover, Materialscientist, McHildinger, MessiFCB, Michael Hardy, Mlewis000, MrOllie, Mwalsh34, Nate, Nbarth, Nealmcb, Niekko1, Niemivh, Niteowlneils, Nixdorf, Nothingofwater, Nubiatech, ObfuscatePenguin, Omarecd, Omicronpersei8, Onceler, Pairadox, Palthainon, Paluchpeter, Pankkake, Pchov, Pgallert, Phil Holmes, Ptewiki, Radagast83, Rfc1394, Rick Sidwell, Rjwilmsi, Roshjack47, Rubiojr, Samriva, Seronline, Shahid789, Sigkill, Sigma 7, Skor, Snowolf, Souliouz, Sreeakshay, Sreya1102, Srleffler, Ssrao08, Stephan Leeds, Storkk, Suruena, Svick, Takanoha, Taral, Teles, Testbells, TheGrimReaper NS, Thejapanesegeek, Tigaente, Tintin1107, Tmaufer, Toe Rag, Tomaxer, Uday, UncleBubba, Vdham, Vjardin, Wbenton, Wbenton-test, Wisden17, Wisq, Wk muriithi, Woohookitty, Wrcarm, Wtmitchell, Wtt, Xpclient, Zamfi, Zr2d2, 424 anonymous edits

Path vector protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=432396364 *Contributors*: Ahruman, Brtbendele, DESiegel, Deanpemberton, Fbriere, JRForster, Jon Awbrey, Juxi, Mauro Bieg, Nealmcb, Shiryaev, Tabletop, 12 anonymous edits

Policy-based routing *Source*: http://en.wikipedia.org/w/index.php?oldid=458480448 *Contributors*: Alba7, Biot, Cmdrjameson, Detach8, EagleOne, Everyking, Five12345, Itai, Jon Awbrey, Kramella2, Mart1 6rille, Mmernex, Outriggr, Pmc, Tecman, Woohookitty, 17 anonymous edits

Routing *Source*: http://en.wikipedia.org/w/index.php?oldid=463995599 *Contributors*: 16@r, 212.150.1.xxx, 777sms, Abdi1543, Adamianash, Aegicen, Afrozenator, AgentPeppermint, Akc9000, Aldie, AlistairMcMillan, Altenmann, Alvestrand, Anna Lincoln, Arthena, Arunrkaushik, Azrael81, Azuris, Baccala@freesoft.org, Bjankuloski06en, Black swallow, Borgx, Brest, Brewhaha@edmc.net, Bruce1ee, Bryan Derksen, Brynosaurus, Callmejosh, ChooseAnother, Ciotog, Cojoco, Conversion script, CosineKitty, D. Recorder, DSRH, Darth Panda, David Lundberg, DavidLevinson, Delaszk, Dgtsyb, Dominic, Dpaehlke, Durty1425, Easyas12c, Echoray, Elfguy, Emperorbma, EvanGrim, Everyking, Extraordinary, GentooBox, Giftlite, Gilliam, Gioto, GoingBatty, Graldensblud, HRV, Haakon, Icelight, Imcdnzl, Informedbanker, Insanity Incarnate, Isidore, Isnow, Itai, JTN, JaGa, Jacob grace, Jadams76, Jec, Jeltz, Jflabourdette, Jnc, Joeldipops, John lindgren, Joseph Dwayne, Joy, Junkblocker, Jwestbrook, Kablammo, Kanags, Karthikittce, Kbrose, Kdineshl, Khattab01, Kjkolb, Kozuch, Kremso, Kvng, Lifefeed, Lilac Soul, LuciusAgrippa, M.duelli, M2petite, MER-C, MLD7865 Auto, Maury Markowitz, Mcbridea, Michael Angelkovich, Michael Hardy, Mindmatrix, Minesweeper, Mmernex, Mmmeg, MrNerdHair, Mwalsh34, N328KF, Nealmcb, NeoJustin, Nethgirb, Neverquick, Nikosdimos, Nmacu, Ocker3, Omarecd, Orange Suede Sofa, Orrc, Oxymoron83, Paul, Pdelong, Pedant17, Pgallert, Phatom87, Phil Christs, Phyzome, PierreAbbat, Piet Delport, Powo, QueenCake, Quidam65, Qwe, R3m0t, Rebounder, Red Thrush, Redsel, Rick Sidwell, RickK, Sevas, Shahid789, Sonett72, SpaceFlight89, Stephenb, Stevenmitchell, Stl, Takanoha, Taxman, Tecman, Thanhantorob, That Guy, From That Show!, The Anome, Thijswijs, Tide rolls, Tjwagner, Tobias Bergemann, Todd Vierling, Towel401, Tregoweth, TripleF, Tristanb, Umreemala, Una Smith, UncleBubba, Unreasonabledude, Unyoyega, Uriah923, Vary, Vineetkrjoshi, Vipulcvyas, Visor, Wadamja, Wbenton, Weregerbil, Wernher, West London Dweller, Weyes, Whocouldibe5, WikHead, Wmahan, Wolfkeeper, Woohookitty, Wrc60, YUL89YYZ, Yama, YordanGeorgiev, Youandme, Youssefsan, ZabMilenko, Zahnradzacken, Zoobee79, 318 anonymous edits

Routing Information Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464892764 *Contributors*: AS, Ahoerstemeier, Ahull, Akc9000, Alan.murray, Alansohn, Aldie, AlephGamma, Alexh19740110, Allolex, Andareed, Andre Engels, Anon lynx, Barberio, Blehfu, Borgx, CSW1986, Callidior, Cybjit, Dajp, DanCasas, Dansen926, David-Sarah Hopwood, Dgtsyb, Dinomite, Druiloor, Drunkenmonkey, Endobson, Enjoi4586, Epbr123, Fab, FlavioMartins, Gamera2, Georgette2, Gmalkin42, Goat-see, Gustavb, H2g2bob, Hadal, Hairy Dude, Hcberkowitz, HeathPetersen, Helon, Honeyman, Iamregin, Iridescent, JHunterJ, JTN, Jec, Jnc, Jon Awbrey, Jwoodger, KD5TVI, Kbrose, Kotiwalo, L33th4x0rguy, Lachlancooper, Lanny68, Lightmouse, Lls71, Maffmommie, Mercury543210, Methossant, Mgeorge27, Michael j harris, Micmon, Minusf, Mmeerman, Mmernex, MrOllie, MureninC, NCurse, Nealmcb, Neilc, Nimiew, Nixdorf, Northernhenge, Oxymoron83, Paul, PaulVIF, Peripitus, Pgallert, Piet Delport, Pmsyyz, Poor Yorick, Ps-cz, R.srinivaas, RJaguar3, Raju5134, RedWolf, Redlazer, Reisio, RexNL, Robert Brockway, Ronz, S.boginskis, Sdrtirs, Seth Nimbosa, Sfisher, Shadowjams, Shahid789, Simon.clayton, Sjc, Skandha101, Suruena, Takanoha, Termininja, ThFabba, ThaddeusB, The Anome, TheGreatFoo, TimonyCrickets, Tomek0001, UncleBubba, Vary, Vickeybhopal, Vjardin, Yaris678, Yashpundir, Zamfi, 245 anonymous edits

Routing protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=453981612 *Contributors*: Akc9000, Aoommen, Arcturus4669, Bbx, Berkay0652, Cntras, ColdthroatEskimo, DARTH SIDIOUS 2, Drgarden, Ecstasyofagony, Fresheneesz, Giga1220, Hcberkowitz, HiDrNick, Ja 62, Jamelan, Jec, KGasso, Kvng, Legolashah, Mercury543210, Mohsinbasharat, Nethgirb, Otrfan, Palbibek, Pgallert, Pip2andahalf, Praveen3053, Prodego, Rahulghose, RedNitrogen, Reedy, Rich Farmbrough, Rwwww, Sharon gngt, Termininja, The Anome, Unfree, Vipulcvyas, Wbenton, Wikipelli, Woohookitty, Yahia.barie, Zodon, 68 anonymous edits

Routing table *Source*: http://en.wikipedia.org/w/index.php?oldid=457931999 *Contributors*: Abune, Akc9000, Alvin-cs, Amigaholic, Andrew D White, Baccala@freesoft.org, Bjankuloski06en, Btilm, CecilWard, David Eppstein, Dungodung, Eddacker, Enduser, Frap, Gabelstaplerfahrer, Gaius Cornelius, Graham87, Harryboyles, Hcberkowitz, Headrush, Helena srilowa, JForget, Jamelan, Kjkolb, Kl4m, Lightmouse, Lysergian, Maher27777, Mckirkus, Michael Hardy, Nethgirb, OrgasGirl, Pdelong, Phatom87, Poco a poco, Prolog, Pyerre, Reclarke, SCriBu, Soumyasch, SpK, Termininja, The Anome, Transhuman7, Twilight Realm, Woohookitty, Wrs1864, Wynnfarm, Zhenqinli, 79 anonymous edits

Split horizon route advertisement *Source*: http://en.wikipedia.org/w/index.php?oldid=462723248 *Contributors*: 7, Adoniscik, AndyKali, Biot, CSW1986, Comatose51, Ed g2s, Edcolins, Eduardo Skywalker, EthernetOverPower, Goatasaur, JayC, Jec, Kbrose, Kwertii, Lukeaw, Mhnin0, Personman, Prashanjitghosh, Rjwilmsi, Ronebofh, Secretlondon, Sfisher, Synchronism, TheParanoidOne, TubularWorld, Wapcaplet, 32 anonymous edits

Domain Name System *Source*: http://en.wikipedia.org/w/index.php?oldid=465443526 *Contributors*: (, 194.17.59.xxx, 28421u2232nfenfcenc, 2D, 4th-otaku, A-moll9, ANONYMOUS COWARD0xC0DE, Abhayakara, Abhinav paulite, Adamantios, Adambiswanger1, Ahoerstemeier, Ahu, Aldie, Ale2006, AlexJP, Alfio, AlistairMcMillan, Allstarecho, Alphachimp, Alphax, Although, Alvestrand, AnK, Anastrophe, Andareed, Andrejj, Andrew Hampe, AndrewN, Andrewpmk, Anna Lincoln, Anthony Appleyard, Anwar saadat, Apacheguru, Arcade, Ardenn, Arichnad, Arifin banjar, Arjayay, Art LaPella, Arvindn, Avoided, Axonizer, Ayla, Azaziel, Azraell, Bahaanet, Balajisarathi, Barek, Bart133, Bayerischermann, Beachy, Beard0, Bedekar.nilesh, Ben-Zin, BenM, Beno1000, Bevo, Bhadani, Bhairab.online, BigMoneyJim, Bigflavor, Binaryspiral, Blanchardb, Bloggeruk, Blue Dot, Bluezy, Boba5fett, Bobblewik, Bobo192, Bobrayner, Bocafan76, Borislav, Bovineone, Brandon, Breno, BrianTung, Brianga, Brichcja, Brucevdk, BryDye, Bryan Derksen, Btharper1221, Byronknoll, Cain Mosni, Calvin 1998, Calvinkrishy, Can't sleep, clown will eat me, CapitalR, Capricorn42, CardinalDan, Cate, Cbuckley, Centrx, Ceyockey, Chairman S., Chasindream, Chealer, Checco, Chenzw, Chipp, Chocolateboy, Chris fluppy, ChrisCork, Chrisdab, Christian List, Cjkporter, ClaretAsh, Cleared as filed, CobraA1, Codes02, Cohesion, Colfer2, Cometstyles, Conversion script, Coredesat, Corti, Cremepuff222, Cst17, Cyanoa Crylate,

Cybjit, Cynric, DGMDGM, DNSstuff, Damian Yerrick, Dandorid, David Haslam, David.Monniaux, Davidshq, Davipo, Dbo789, Dcsutherland, Dee.tee, Delivery:435, Demitsu, Dennis Brown, DerHexer, Derat, Diannaa, Discospinster, Dithridge, Djkrajnik, Dman727, Dolda2000, DoomBringer, Doomdayx, Dougalf, Dr.queso, Draicone, Dreg mogul, Dripdk, Drj, Drjan, Drowne, Dtobias, Dwheeler, Dysprosia, EdJohnston, Editor plus, Edward K, Edward321, Egmontaz, Ehn, Elinruby, Eloquence, Elvey, Emdx, Epastore, Epbr123, Ephillips, Equendil, Eras-mus, Eric Kvaalen, ErikWarmelink, Evil Monkey, Explicit, Feezo, Fenitharbour, Feydey, Fifthnail, FisherQueen, FleetCommand, Fleminra, Flubbit, Forseti, Fortunecookie289, Frap, Fred Gandt, Fudoreaper, Funkysh, Funnyfarmofdoom, FvdP, Fæ, GFellows, Garas, Gary King, Gdr, Geoffspear, Gerbrant, Gerixau, Gilesmorant, Glane23, Glenn, Glrx, Gobbleswoggler, Gogo Dodo, GoingBatty, Graciella, GraemeL, Graham87, Grayshi, Greg.smith.tor.ca, Groogle, Guanxi, Gurch, Gutzalpus, Haakon, Hadal, Hairy Dude, HalJor, Hamtechperson, Hardaker, Harris7, Hayabusa future, Hbatra, Hcberkowitz, Hdt83, Helix84, Helixblue, Henriwatson, Honta, Horsten, Htaccess, Hu12, IJMacD, Ibuzzo, Idont Havaname, Insanity Incarnate, Inter, InverseHypercube, Iohannes Animosus, Ispabierto, Iulianu, J.delanoy, JForget, JH-man, JSGould, JTN, Jackbrownwan, Jake Wartenberg, Jamyskis, Jarsyl, Jasynnash2, Jauerback, Javawizard, Jawsper, Jcw69, Jdthood, Jec, Jfpierce, Jgeer, Jh51681, Jhasai, Jhawkinson, Jiddisch, Jinlye, Jivecat, Jj137, Jlaidman, Jmcw37, Jnc, JoanneB, Joe Sewell, Jog1973, Johan Elisson, John Broughton, John Vandenberg, JohnSmith777, Johnarett, Jonah S., Joonas Govenius, Joseph Dwayne, Joseph Solis in Australia, Jpatokal, JuJube, Jy, Jyoshimi, Jyoujjawal, KGasso, KLLvr283, Karada, Karengpve, Kateshortforbob, Katieh5584, Kbrose, Kelly Martin, Kenguest, Kinema, Kingpin13, Kinneyboy90, Kitplane01, Kizor, Kjdunlap, Kjetil r, KnightRider, Kop, Kraftlos, Krawi, Krellis, Krooga, Kungfuadam, Kurumban, Kvng, Kwamikagami, Kwertii, Labongo, Lambtron, Leandrod, Lee Daniel Crocker, Lemming, Lentoyip, LeoNomis, Liamdaly87, Liftarn, Lightdarkness, Lightmouse, Limajean34983, LimoWreck, LionKimbro, Lod, Lotje, Ludde23, Luna Santin, Lupin, Lysdexia, MC MasterChef, MER-C, MWD27, Maestro25, Mahanga, Mam711, Mange01, Marc Mongenet, Marcel van b, Marcod'Itri, Marechal Ney, Mark Bergsma, Mark7-2, Materialscientist, Matilda, Matt Darby, Matt.smart, MattGiuca, Matthewvelie, Mattias.Campe, Mattrobinson78, Max Cheung, Maximillion Pegasus, Mendel, Merlion444, Merovingian, MetsFan76, Mfwitten, Mgechert, Michael Hardy, Michel Jansen, Mike Rosoft, Mindmatrix, Minesweeper, Minghong, MonRa, Moondyne, Moonriddengirl, Moskvax, MrJones, MrKris, Mrguyguy226, Mrzaius, Msantcroos, Msjadex, Mwtoews, Mysekurity, N0YKG, Nabber00, Nakon, Name99, Nanshu, Naraht, Narayanacharan, Natkeeran, NawlinWiki, Nburden, Nealcardwell, Neil916, Netesq, Netkinetic, NevilleDNZ, Nevlino, NewEnglandYankee, Niceguyedc, NickCT, NickW557, Nickmaine, Nilesh2293, Nimiew, Niteowlneils, Nivix, Nneonneo, NotAnonymous0, Notheruser, Nubiatech, O, Oddity-, Olddocks, Oldwes, Oliver341, Omegagod, Omicronpersei8, Omiksemaj, Omniplex, Openhazel, Optimist on the run, Oshah, Oskar Sigvardsson, Ott2, Ottawahitech, Oxymoron83, Ozruta, Pagingmrherman, Pak21, Parsecboy, Patrick, Paul Stansifer, Paulson74, Pcb21, Pedant17, Peter Karlsen, Peterbrown1, Pgan002, Pharaoh of the Wizards, Phatom87, Philarete, Philip Trueman, PhilipMW, Piano non troppo, Pierre Monteux, Pilom, Pinethicket, Piotras, Pjrich, Pointer1, Poweroid, Puffin, Qasimalikhawaja, QmunkE, Quarl, Quincywalltech, Quintote, Qwertyca, Qwertyus, R4f, RAM, Ragib, RainbowOfLight, Rameshbabu.itian, RandomStringOfCharacters, Rapomon, Rasbelin, Rasmus Faber, Ravedave, Reach Out to the Truth, RedHillian, Reeser8, Regibox, ReiDx, RetiredUser2, Rfc1394, Riana, Rich Farmbrough, Rick Block, Rick Sidwell, Rikkus, Rjd0060, Rjgodoy, Rob Hooft, Robertb-dc, Roger Davies, Ron Ritzman, Ronz, Rottenblackberry, Roughana, RoyBoy, Rrius, Rror, Rshin, RussBlau, Ruwanindika, Rwashi, S3000, SDC, SJP, SMC89, ST47, SWAdair, Salokha, Samboy, Sanbio, Sanjiv swarup, Scarian, Sceptre, Schmloof, Schnolle, Science4sail, Scolobb, Scott.somohano, Scratchy, Seaphoto, Sen86, Serezniy, Serie, Shadowhillway, Shadowjams, Shalom Yechiel, Shams9598, Shenme, Shentino, Shii, Simigh, Simon Dodd, Simonjonzie, Sionus, SirPavlova, Skarebo, Sleigh, Smalljim, Smallpond, Smappy, Smashville, Smichae, Snowolf, Snoyes, Socialservice, Sokool, SolarisBigot, Solveforce, Some jerk on the Internet, Sonomadiver@gmail.com, Sonuahluwalia, SpaceFlight89, Spellmaster, SperoSpes, Spinboy, Spishco, Sstrader, StaticGull, StealthFox, Stephan Leeds, Stephen Gilbert, Stib, Stwalkerster, SuperHamster, Suruena, SusanLarson, Sweet473, SyntaxError55, TAG.Odessa, Takis, Tardis, Taw, TedPavlic, Tedder, Teppic, The Anome, The King's Peach, The Thing That Should Not Be, The locster, Thefunkygibbon, Thumperward, Tide rolls, Tim1988, Timwi, Tirerim, Tobias Bergemann, Tobias Conradi, Tonsofpcs, Tony1, Torzsmokus, Totty3478, Towel401, Trantjd, Tree Biting Conspiracy, Tregoweth, Trusilver, Ttony21, Tualha, Turnstep, Twajn, Twinxor, Tylerdmace, Tyomitch, Ufirst, Unara, Uncle G, UncleBubba, Updatebjarni, UrsaFoot, Use Bombs, Valenciano, Vanished user 39948282, Vanisheduser12345, Vary, Vera Cruz, Versageek, VictorianMutant, Vivio Testarossa, Vldhcp, Vocaro, Vrenator, Vssun, W2bh, WRJ, Wasisnt, Wavelength, Wayward, Webmaster4india, Webzero, Weike9009, Wesley, Weyes, Wigren, Wiki alf, Wiki104, WikiReviewer.de, Wikibase, Wikibob, Wikibofh, Wikiborg, Wikijeff, Wikipedia.org@parkerpress.com, WikipediaEdit, Wikipelli, Will Beback Auto, Wk muriithi, Wknight94, Wmahan, Wrs1864, Xardion, Xionbox, Yamamoto Ichiro, Yes-Chef?, Yogi de, Yurik, Yyy, ZSBoS, ZaffaNE, Zelphar, Zfr, Zidane2k1, Zro, Zundark, Zzuuzz, Пика Пика, 1697 anonymous edits

Dynamic Host Configuration Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=464756877 *Contributors*: 78.26, 909078L, Abhayakara, Abhi3385, Abisys, Abune, Acroterion, AdamJudd, Adoniscik, Afiler, Aitias, Alaniaris, Alansohn, Aldie, Alex rosenberg35, Alfio, Alison22, AlistairMcMillan, Allstarecho, Almightylinuxgod, Alphachimp, Alvin-cs, Amaccuish, Ampsarus, Andareed, Andre Engels, Andrei Stroe, Animum, Anuragkrgupta, Apokrif, Arichnad, Arkrishna, Ashakiran, Ashwin, Atomician, BL, BW, Bardo, BartRoos, BenAveling, Billdorr, Bobby D. DS., Bogdangiusca, Bookbrad, BrianOfRugby, Brucefulton, CALR, Can't sleep, clown will eat me, Celarnor, Celinama, Centrx, CharlieEchoTango, Chealer, Chris 73, Chris the speller, Chrumps, ClanCC, Cleared as filed, Cliffb, Closedmouth, Coastalsteve984, Conversion script, Coutcin, Ctmt, DARTH SIDIOUS 2, DKEdwards, DStoykov, Danger, Daniel.Cardenas, Davee, Davidkazuhiro, Davidoff, Davidstrauss, Dbu, DeadEyeArrow, Dee Jay Randall, Denelson83, Desiremore, Dgies, DigitalSorceress, Dols, Dreadstar, Dremora, Dylan Lake, EagleOne, Earendur, Echosmoke, Edward, Egjose, Ehn, Electrified mocha chinchilla, Elwell, Enjoi4586, ErikWarmelink, Evil Monkey, Ewlyahoocom, Feureau, Flash.killer, Fleung, Friday, Fudoreaper, Furrykef, G Sisson, Gaius Cornelius, Gamera2, Gareth Owen, Gary King, Giftlite, Gimboid13, Grafen, Grenavitar, Guru cool, Gwalker nz, Hackingyourdna, Hadal, Hanjifi, Hannes Hirzel, HarisM, Harshbhai, Hazphi, Hazzamon, Hcberkowitz, Heathbar5477, Helix84, Hephaestos, Hide1713, Hughcharlesparker, Hughey, Infofarmer, Ironywrit, IsUsername, JTN, JaGa, Jasoltape, JasonWoof, Jimsve, Joeinwap, Joerg Reiher, Josh Parris, Joy, Jrvz, Jude Rosario, Karora, Kbrose, Kentyman, Kevinmon, Kgfleischmann, Kinema, King Toadsworth, Kingpin13, Kinu, Krellis, Kubigula, Kurykh, Kvng, LAAFan, Labongo, Lacen, Lahiru k, Lightmouse, Lkinkade, Looxix, Loren.wilton, Lowellian, M4gnum0n, M7, MER-C, Mahewa, Makelelecba, Mam711, Mandarax, Mani1, Materialscientist, MattGiuca, Matthuxtable, Maximbo, Mcgonrya, Mechanical digger, Memming, Michael Devore, Mike Dillon, Mike Rosoft, Mindmatrix, Ministry of Truth, Mking30, Moeron, Moondyne, Mr Stephen, MrExplosive, Msabramo, Mwtoews, NCdave, Nachmore, Nafmosaved, Nastajus, Naved 12, Netsnipe, Ngriffeth, Nichtich, Nikai, Nimiew, Nixdorf, Njroberts, Nnp, Nubiatech, Nutster, Nv8200p, Ocker3, Ocram, Ojw, Omniplex, One, Ootachi, PCHS-NJROTC, PRBryson, PV=nRT, Pablicosta, Pagingmrherman, Pandemic, Parag2010, Parbatyogesh, Paris.butterfield, PeaceNT, Pedant17, Peng, Phatom87, Philcha, Piet Delport, Pinkadelica, Pion, Prohlep, Pseudomonas, QRX, Quarl, RJaguar3, Raanoo, Radiant chains, Raghav, RattleMan, Razimantv, Rchandra, RedWolf, Reub2000, Rfc1394, Rich Farmbrough, Richwales, Rikboven, Rjstott, Rjwilmsi, Rl201, Rob Hooft, Rodrigo.haus, Rrburke, Rursus, SJP, Sam Korn, Sameer.sujeet, Schaea, Scott.somohano, ScottSteiner, Sdrtirs, Sesshomaru, Shady69, Shields020, Shiftoften66, Shiryaev, Skyfiler, Snarius, SpK, Spasemunki, SpeedyGonsales, SportWagon, Srbanator, Srharriott, Stephan Leeds, Stevenm.ict, Suburbanslice, Suffusion of Yellow, Superborsuk, Superm401, Suruena, Swatithorve, TYelliot, Taochen, Taoqirkhosa, Tcb, Teemuk, Tellyaddict, Template namespace initialisation script, Teraya, The Thing That Should Not Be, The.valiant.paladin, Theking17825, Thomas d stewart, Thumperward, Tide rolls, Tim.sylvester, Tothwolf, Totty3478, Tremilux, True Pagan Warrior, Trusilver, Tyler.szabo, Ummit, UncleBubba, Utcursch, VernoWhitney, Vijeshchandran, Vocaro, Voyagerfan5761, Waggers, Wasisnt, Wdrazo, Wereon, Wiki104, Wikisierracharlie, Wilson.canadian, Winterst, Wisco, Wisher, Wizzard2k, Wk muriithi, Wrs1864, XLiquidIceX, Xhienne, Xionbox, Xoxon1kk1, Yesteraeon, YordanGeorgiev, Yuckfoo, Zac439, Zedla, ^demon, 858 anonymous edits

Hypertext Transfer Protocol *Source*: http://en.wikipedia.org/w/index.php?oldid=465371679 *Contributors*: 1234r00t, A!eX, A-moll9, Aapo Laitinen, Abdull, Abu ali, Acdx, Adoniscik, Agawish, Agupte, Ahoerstemeier, Aitias, Alansohn, Aldie, AlefZet, Alerante, Ali@gwc.org.uk, AlistairMcMillan, Amalthea, Andareed, Andkim99, Andre Engels, Android Mouse, Angus Lepper, Anna Lincoln, AnotherSprocket, Anwar saadat, Apparition11, Arch o median, Ash, Ashley Y, Asj123fds, Asteiner, Astroview120mm, Atemperman, Audriusa, Avb, Avnjay, Baa, Babbage, Babedacus, Babysabre, Bamkin, BaronLarf, Basil.bourque, BeakerK44, Beefyt, Beinder, Beland, Ben-Zin, BenBildstein, Benandorsqueaks, Benanhalt, Bernfarr, Betacommand, Betterworld, Bevan7, Bevo, Bg, Bgwwlm, Bihco, Bjankuloski06en, Blade44, Blanchardb, Blm, Bluemask, Bluerasberry, Bobo192, Bongwarrior, Booyabazooka, Bpantalone, Brian.fsm, Brownsteve, Bsrdjan, Bull3t, C. A. Russell, C.Fred, CWii, Callmederek, Calor, Caltas, Calvin 1998, Cameltrader, Can't sleep, clown will eat me, Canderson7, Capricorn42, CardinalDan, Catamorphism, Cburnett, CecilWard, Cedy, Chadfiller, Charles Gaudette, Chasingsol, Checking, Chenqiang.prc, ChongDae, Christian75, Chruck, Coasting, Conversion script, Coroberti, Corpx, Cryptic, Cwolfsheep, Cyberstrike3000X, CyrilB, DKEdwards, DVdm, Daf, Dalitvoice, Damian Yerrick, Damnreds, DanBLOO, DanBishop, Danculley, Daniboydl07, Danielfolsom, DasRakel, Davidjk, Davilima, Dawn Bard, Dcoetzee, Ddas, De728631, DeadEyeArrow, Delicates, Delirium, DerBorg, DerHexer, DigitalNinja, Dionyziz, Discospinster, Dmyersturnbull, Doria, Dr bab, Drcwright, Dspradau, ENeville, Earthlyreason, Edward, Ehn, El C, ElKevbo, Electron9, Eli the Bearded, Elimerl, Ellywa, Elving, Englishrose, Enjoi4586, Ent, Epbr123, EphemeralJun, Equendil, Esanchez7587, Evil Monkey, Ewawer, Extransit, Eyreland, Femto, Ferkelparade, Firetrap9254, Flockmeal, Frap, Fred Bradstadt, Furrykef, Fuzzygenius, GJHC1, GW Simulations, Gaccolin, Gaius Cornelius, Gamernotnerd, Gdo01, Gen. von Klinkerhoffen, Gfoley4, Giftlite, GimliDotNet, Godtrog, GordonMcKinney, Gracenotes, Graciella, Graham87, Grantglendinning, Gregkaye, Grouse, Gscshoyru, Gsiegman, Gurchzilla, Guy Harris, Haakon, Habbabuba91, Haiiiiiillsatana, Ham Pastrami, Hans Genten, HappyCamper, Happysailor, HarisM, Haseo9999, Hashar, Hashproduct, Helix84, HexaChord, Horsten, Hrafnkell.palsson, Hrvoje Simic, Htonl, Hu12, Hydrogen Iodide, Hyperthermia, IMSoP, IShadowed, Icairns, Idyllic press, Ihatejava, Iluvcapra, Imnotminkus, Imroy, Incnis Mrsi, InfoRetrieval, Interiot, Intgr, Irrypride, Itpastorn, Ivank06, Ivko, Ixfd64, J.delanoy, JHolman, JTN, Jaan513, Jacobolus, JamesBWatson, Jarkospratt, Jarkspratt, Jdowland, Jeffq, Jeltz, Jennica, Jleedev, Jmar777, John Quiggin, John Vandenberg, John of Reading, Jojhutton, Jonathan O'Donnell, Jonnabuz, JonoF, Josh3736, Joyous!, Jrobinjapan, Jstrater, Jusdafax, JustAnotherJoe, KAtremer, KGasso, Ka-Ping Yee, Karingo, Karl Dickman, Kbrose, Kesac, Kesla, Kgfleischmann, Kinema, Kiwi128, Knutux, Koavf, Kocio, Konniret3, Korealin, KrakatoaKatie, Krellion, Krellis, Ksn, Kundor, Kungfuadam, KurtRaschke, Kvng, Kwi, Kylu, Kymacpherson, La Parka Your Car, Lady Serena, Lannm, Leafyplant, Lee Carre, Lee J Haywood, Lethe, Lezek, Lightdarkness, Lilac Soul, Linusnk, Lloydpick, Lotje, Lotu, Lproven, Luna Santin, Lupinoid, Lysdexia, M4gnum0n, MLauba, Mabdul, Macademe, MackSalmon, Madhero88, Maebmij, Magaman dude, Magioladitis, Mange01, Marek69, Martin BENOIT, Marx01, Materialscientist, Mathiastck, Mdd, Mdf, Meekohi, Mehfuz, Meneth, MikeF, Mikerlynn, Mindmatrix, Mjb, Mlibby, Mmj, Mnot, Modster, Mohammedsheikh786, Mormegil, Moustafaza, Mpeylo, Mr Stephen, MrExplosive, MsDivagin, Mschel, Mumia-w-18, Mwarren us, Mwtoews, Nanshu, NawlinWiki, Negrulio, Nevyn, NewEnglandYankee, Newone, Nezzadar, Nicolas1981, Nigelj, Nightraider0, Nightstallion, Nikola Smolenski, Ninly, Niqueco, Nixdorf, Nixeagle, Nknight, Nlu, No Guru, Nubiatech, Nurg, Nyelvmark, ObfuscatePenguin, Okyea, Omniplex, Oneiros, Optiguy54, Orbnauticus, OsoLeon, Oxymoron83, Papadakis2007, Part Deux, Pascal666, Patrick, Paul Mackay, Paul-L, Pb30, Persian Poet Gal, Petero9, Pforret, Phatom87, Philip Trueman, Piano non troppo, Pioneerking, Pip2andahalf, Praefectorian, Produke, Psychonaut, Pxma, Qff828, R3m0t, RHaworth, RJaguar3, RainbowOfLight, Rainman-sr, Raise exception, RandomHumanoid, Ravik, Reach Out to the Truth, RedWolf, Reschke, Retiono Virginian, RexNL, Rhaskallion, Rich Farmbrough, Rischmueller, Rjwilmsi, Rmhermen, Rmosler2100, RobMattson, Robert Xia, RockMFR, Ronhjones, RossPatterson, Rror, RyanCross, Ryulong, SJP, SWAdair, Sae1962, Saebjorn, Sakkath, Salvio giuliano, Samuel, Sander Säde, Saoshyant, Saturn star, SchnitzelMannGreek, Schwallex, Scientus, Sdrtirs, Sean Whitton, Seb-Gibbs, Sekro, Shadowjams, Sharon08tam, SheeEttin, Shiftoften66, Sietse Snel, SiobhanHansa, SixSix, SkE, Socrates2008, SoftX, Spaceman85, Spandrawn, Sparky132, SqueakBox, Sri Krishna Geova Allah, Sses401, Stephan Leeds, StephenBuxton, Stephenb, SteveLoughran, Stevenup7002, Stolee, Stupid Corn, Suppie, Suruena, Syrthiss, T-borg, THEN WHO WAS PHONE?, Tabledhote, TakuyaMurata, Tbhotch, TechTony, Template namespace initialisation script, Tempodivalse, TenOfAllTrades, Th.matt.wiki, The Anome, The Thing That Should Not Be, The Utahraptor, TheJosh, Think777, Thumperward, Tide rolls, Timc1212, Timmywimmy, Timneu22, Tlesher, Tobias Bergemann, Tobias382, Todd Vierling, Tomaxer, Tommy2010, TonyAiuto, Toto Mommam, Tpbradbury, Tpirfan28, Tregoweth, Trusilver, TuukkaH, UU, Uncle Dick, Uncle G, UncleBubba, Universalss, Universimmedia, Uruiamme, Utcursch, VMS Mosaic, Versus22, VictorAnyakin, Vinoth.t, Voxii, Wayne Hardman,

Wikipe-tan, Wikipelli, Wintonian, Wizzard2k, Wolfomatic5, Woohookitty, Wtmitchell, Yacht, Yaronf, Yomcat, Yonatan, Yuhong, Zachlipton, Zalnas, Zvn, ^demon, Île flottante, 1201 anonymous edits

Image Sources, Licenses and Contributors

File:Internet layering.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Internet_layering.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jsoon eu (talk)

File:Message flows.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Message_flows.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jsoon eu (talk)

File:Message flows and Routing.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Message_flows_and_Routing.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jsoon eu (talk)

File:Protocol and Software layering.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Protocol_and_Software_layering.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jsoon eu (talk)

File:Internet map 1024.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Internet_map_1024.jpg *License*: Creative Commons Attribution 2.5 *Contributors*: Barrett Lyon The Opte Project

File:Distributed Processing.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Distributed_Processing.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Bp2010.hprastiawan

Image:NETWORK-Library-LAN.png *Source*: http://en.wikipedia.org/w/index.php?title=File:NETWORK-Library-LAN.png *License*: Creative Commons Attribution 3.0 *Contributors*: Hcberkowitz

File:EPN Frame-Relay and Dial-up Network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:EPN_Frame-Relay_and_Dial-up_Network.svg *License*: Creative Commons Attribution-Share Alike *Contributors*: Ludovic.ferre

File:Virtual Private Network overview.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Virtual_Private_Network_overview.svg *License*: Creative Commons Attribution-Share Alike *Contributors*: Ludovic.ferre

File:Network Overlay.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_Overlay.svg *License*: Creative Commons Attribution-Share Alike *Contributors*: Ludovic.ferre]]

Image:PD-icon.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:PD-icon.svg *License*: Public Domain *Contributors*: Alex.muller, Anomie, Anonymous Dissident, CBM, MBisanz, Quadell, Rocket000, Strangerer, Timotheus Canens, 1 anonymous edits

File:OSI-model-Communication.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:OSI-model-Communication.svg *License*: Public Domain *Contributors*: Runtux

Image:IP stack connections.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:IP_stack_connections.svg *License*: GNU Free Documentation License *Contributors*: en:User:Kbrose

Image:UDP encapsulation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:UDP_encapsulation.svg *License*: GNU Free Documentation License *Contributors*: en:User:Cburnett original work, colorization by en:User:Kbrose

Image:ARP Spoofing.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:ARP_Spoofing.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:0x55534C

Image:IBM Turboways ATM 155.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:IBM_Turboways_ATM_155.jpg *License*: Public Domain *Contributors*: Original uploader was Rjamorim at en.wikipedia

Image:Checksum.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Checksum.svg *License*: Public Domain *Contributors*: Original by Helix84, adapted by Jorge Stolfi

Image:Ethernet RJ45 connector p1160054.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ethernet_RJ45_connector_p1160054.jpg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: User:David.Monniaux

File:Loudspeaker.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Loudspeaker.svg *License*: Public Domain *Contributors*: Bayo, Gmaxwell, Husky, Iamunknown, Mirithing, Myself488, Nethac DIU, Omegatron, Rocket000, The Evil IP address, Wouterhagens, 16 anonymous edits

File:10Base5transcievers.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:10Base5transcievers.jpg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Robert.Harker at en.wikipedia

Image:Network card.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_card.jpg *License*: GNU Free Documentation License *Contributors*: User:Nixdorf

File:Network switches.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Network_switches.jpg *License*: unknown *Contributors*: ShakataGaNai

File:Coreswitch (2634205113).jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Coreswitch_(2634205113).jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Dave Fischer

Image:4 port netgear ethernet hub.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:4_port_netgear_ethernet_hub.jpg *License*: Public Domain *Contributors*: Hohum, JackPotte, Plugwash, 1 anonymous edits

Image:Frame relay.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Frame_relay.jpg *License*: Public Domain *Contributors*: Ruwanindika

Image:Frame Synced Stream.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Frame_Synced_Stream.jpg *License*: GNU Free Documentation License *Contributors*: Erin Frye

Image:PCM Stream.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:PCM_Stream.jpg *License*: GNU Free Documentation License *Contributors*: Erin Frye

Image:Commutation.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Commutation.jpg *License*: GNU Free Documentation License *Contributors*: Erin Frye

File:Juniper srx210 front view.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Juniper_srx210_front_view.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: MarcPG

File:SRI First Internetworked Connection diagram.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:SRI_First_Internetworked_Connection_diagram.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Russavia

File:SRI Packet Radio Van.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:SRI_Packet_Radio_Van.jpg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Russavia

File:Multipexing demultiplexing scheme en.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Multipexing_demultiplexing_scheme_en.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Computer.svg: Tango Desktop Project Esquema-proxy-internet.svg: Randomicc Gnome-modem.svg: GNOME icon artists derivative work: Ggia (talk)

File:Frequenzmultiplex001.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Frequenzmultiplex001.jpg *License*: GNU Free Documentation License *Contributors*: Matthias Bock Original uploader was Matthias Bock at de.wikipedia

File:Telephony multiplexer system.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Telephony_multiplexer_system.gif *License*: Creative Commons Attribution *Contributors*: Tony R. Kuphaldt

Image:telecommunication-multiplexing.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Telecommunication-multiplexing.gif *License*: GNU Free Documentation License *Contributors*: Joey-das-WBF, Paxse, Poweroid, Psychonaut, Remember the dot, 1 anonymous edits

Image:Smartswitch6000.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Smartswitch6000.jpg *License*: GNU Free Documentation License *Contributors*: User:Matthäus Wander

Image:24-port 3Com switch.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:24-port_3Com_switch.JPG *License*: Public Domain *Contributors*: Rodrigo César

File:Linksys48portswitch.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Linksys48portswitch.jpg *License*: Attribution *Contributors*: Original uploader was Akc9000 at en.wikipedia

File:Switch-and-nest.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Switch-and-nest.jpg *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was J.smith at en.wikipedia

Image:PPPPhaseDiagram.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:PPPPhaseDiagram.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Jesse Viviano

Image:Pdu and sdu.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Pdu_and_sdu.svg *License*: Public Domain *Contributors*: Mik81

File:Ciscosystemsrouteratcern.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ciscosystemsrouteratcern.jpg *License*: GNU Free Documentation License *Contributors*: Coolcaesar

File:!Adsl connections.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:!Adsl_connections.jpg *License*: GNU Free Documentation License *Contributors*: Asim18

File:OpenWRT 8.09.1 LuCI screenshot.png *Source*: http://en.wikipedia.org/w/index.php?title=File:OpenWRT_8.09.1_LuCI_screenshot.png *License*: GNU General Public License *Contributors*: Moxfyre (talk)

File:Leonard-Kleinrock-and-IMP1.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Leonard-Kleinrock-and-IMP1.png *License*: Public Domain *Contributors*: Leonard Kleinrock

File:ERS-8600.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:ERS-8600.JPG *License*: GNU Free Documentation License *Contributors*: Original uploader was PassportDude at en.wikipedia

Image:Spanning tree protocol at work 1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_1.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

Image:Spanning tree protocol at work 2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_2.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

Image:Spanning tree protocol at work 3.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_3.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

Image:Spanning tree protocol at work 4.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_4.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

Image:Spanning tree protocol at work 5.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_5.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

Image:Spanning tree protocol at work 6.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Spanning_tree_protocol_at_work_6.svg *License*: Creative Commons Attribution 3.0 *Contributors*: GhosT

image:broadcast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Broadcast.svg *License*: Public Domain *Contributors*: Easyas12c

image:cast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Cast.svg *License*: Public Domain *Contributors*: Easyas12c

image:anycast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Anycast.svg *License*: Public Domain *Contributors*: Easyas12c, H Padleckas, Jarekt, 1 anonymous edits

image:multicast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Multicast.svg *License*: Public Domain *Contributors*: Easyas12c, Lupo, 1 anonymous edits

image:unicast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Unicast.svg *License*: Public Domain *Contributors*: Easyas12c, Perhelion

image:geocast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Geocast.svg *License*: Creative Commons Zero *Contributors*: Revolus

Image:Internet_map_in_February_82.jpg144kB.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Internet_map_in_February_82.jpg144kB.png *License*: Public Domain *Contributors*: w:Jon PostelJon Postel

File:IP Address Match.png *Source*: http://en.wikipedia.org/w/index.php?title=File:IP_Address_Match.png *License*: Public Domain *Contributors*: Baccala@freesoft.org, It Is Me Here, The wub, TommyBee, WikipediaMaster, Zapyon, 2 anonymous edits

File:CIDR Address.png *Source*: http://en.wikipedia.org/w/index.php?title=File:CIDR_Address.png *License*: Public Domain *Contributors*: Baccala@freesoft.org, Davepape, Longbow4u, WikipediaMaster, 2 anonymous edits

Image:IGMP_basic_architecture.png *Source*: http://en.wikipedia.org/w/index.php?title=File:IGMP_basic_architecture.png *License*: Public Domain *Contributors*: Harpreet Eighty-Two

File:IP stack connections.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:IP_stack_connections.svg *License*: GNU Free Documentation License *Contributors*: en:User:Kbrose

File:UDP encapsulation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:UDP_encapsulation.svg *License*: GNU Free Documentation License *Contributors*: en:User:Cburnett original work, colorization by en:User:Kbrose

Image:Ipv4 address.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv4_address.svg *License*: Public Domain *Contributors*: w:en:User:IndeterminateIndeterminate

Image:Ipv6 address.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv6_address.svg *License*: Public Domain *Contributors*: Indeterminate

File:IP in IP Encapsulation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:IP_in_IP_Encapsulation.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Cybjit

Image:Ipv6 address leading zeros.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv6_address_leading_zeros.svg *License*: Public Domain *Contributors*: Ipv6_address.svg: Indeterminate derivative work: BobbyPeru (talk)

File:Ipv6 header.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Ipv6_header.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Mro

Image:multicast.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Multicast.svg *License*: Public Domain *Contributors*: Easyas12c, Lupo, 1 anonymous edits

Image:Full Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Full_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Restricted Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Restricted_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Port Restricted Cone NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Port_Restricted_Cone_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

Image:Symmetric NAT.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Symmetric_NAT.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Christoph Sommer

File:EISA TokenRing NIC.JPG *Source*: http://en.wikipedia.org/w/index.php?title=File:EISA_TokenRing_NIC.JPG *License*: Public Domain *Contributors*: Original uploader was Wbenton at en.wikipedia

File:Subnetting operation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Subnetting_operation.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Gerson Chicareli

Image:Random Early Detection algorithm en.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Random_Early_Detection_algorithm_en.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: helix84

Image:Sliding Window.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Sliding_Window.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Alexander Krivács Schrøder (DarkPhoenix)

Image:Tcp state diagram fixed new.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Tcp_state_diagram_fixed_new.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Sergiodc2, Marty Pauley, Scil100

Image:Tcp.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Tcp.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Mike de

Image:BGP FSM.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:BGP_FSM.svg *License*: GNU Free Documentation License *Contributors*: Johannes Rössel (talk)

File:BGP Table growth.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:BGP_Table_growth.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Mro

File:Internet AS.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Internet_AS.svg *License*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors*: Mro

Image:Dijksta Anim.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Dijksta_Anim.gif *License*: Public Domain *Contributors*: Ibmua

Image:Dijkstras_progress_animation.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Dijkstras_progress_animation.gif *License*: Creative Commons Attribution 3.0 *Contributors*: Subh83

image:networkabcd.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Networkabcd.svg *License*: Public Domain *Contributors*: Original uploader was Lembut at en.wikipedia

Image:Flood.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Flood.gif *License*: unknown *Contributors*: SuperFLoh

Image:FloodAck.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:FloodAck.gif *License*: unknown *Contributors*: Bukk, Jahobr, SuperFLoh

File:Hop-count-trans.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Hop-count-trans.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Stagira

Image:A-B-C.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:A-B-C.svg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Created by Biot

Image:Domain name space.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Domain_name_space.svg *License*: Public Domain *Contributors*: Bjankuloski06en, GeorgHH, Ggia, Herbythyme, LionKimbro, Ma-Lik, Mdd, Waldir, 12 anonymous edits

Image:An example of theoretical DNS recursion.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:An_example_of_theoretical_DNS_recursion.svg *License*: Public Domain *Contributors*: LionKimbro, The Anome, Waldir, 4 anonymous edits

Image:DNS in the real world.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:DNS_in_the_real_world.svg *License*: Public Domain *Contributors*: LionKimbro, Waldir, 2 anonymous edits

File:DHCP Server.png *Source*: http://en.wikipedia.org/w/index.php?title=File:DHCP_Server.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Patpat

Image:Http request telnet ubuntu.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Http_request_telnet_ubuntu.png *License*: Public Domain *Contributors*: TheJosh

License

Creative Commons Attribution-Share Alike 3.0 Unported
//creativecommons.org/licenses/by-sa/3.0/