

# Info I

## Protocolos de Red

# Contents

## Articles

IEEE 802	1
Ethernet	3
Ethernet frame	10
IPv4	15
User Datagram Protocol	25
Transmission Control Protocol	31
OSI model	48
Internet protocol suite	57

## References

Article Sources and Contributors	68
Image Sources, Licenses and Contributors	71

## Article Licenses

License	72
---------	----

# IEEE 802

**IEEE 802** refers to a family of IEEE standards dealing with local area networks and metropolitan area networks. More specifically, the IEEE 802 standards are restricted to networks carrying variable-size packets. (By contrast, in cell relay networks data is transmitted in short, uniformly sized units called cells. Isochronous networks, where data is transmitted as a steady stream of octets, or groups of octets, at regular time intervals, are also out of the scope of this standard.) The number 802 was simply the next free number IEEE could assign,<sup>[1]</sup> though “802” is sometimes associated with the date the first meeting was held — February 1980.

The services and protocols specified in IEEE 802 map to the lower two layers (Data Link and Physical) of the seven-layer OSI networking reference model. In fact, IEEE 802 splits the OSI Data Link Layer into two sub-layers named Logical Link Control (LLC) and Media Access Control (MAC), so that the layers can be listed like this:

- Data link layer
  - LLC Sublayer
  - MAC Sublayer
- Physical layer

The IEEE 802 family of standards is maintained by the IEEE 802 LAN/MAN Standards Committee (LMSC). The most widely used standards are for the Ethernet family, Token Ring, Wireless LAN, Bridging and Virtual Bridged LANs. An individual Working Group provides the focus for each area.

## Working groups

Name	Description	Note
IEEE 802.1	Bridging (networking) and Network Management	
IEEE 802.2	LLC	inactive
IEEE 802.3	Ethernet	
IEEE 802.4	Token bus	disbanded
IEEE 802.5	Defines the MAC layer for a Token Ring	inactive
IEEE 802.6	MANs (DQDB)	disbanded
IEEE 802.7	Broadband LAN using Coaxial Cable	disbanded
IEEE 802.8	Fiber Optic TAG	disbanded
IEEE 802.9	Integrated Services LAN (ISLAN or isoEthernet)	disbanded
IEEE 802.10	Interoperable LAN Security	disbanded
IEEE 802.11 a/b/g/n	Wireless LAN (WLAN) & Mesh (Wi-Fi certification)	
IEEE 802.12	100BaseVG	disbanded
IEEE 802.13	Unused <sup>[2]</sup>	
IEEE 802.14	Cable modems	disbanded
IEEE 802.15	Wireless PAN	
IEEE 802.15.1	Bluetooth certification	
IEEE 802.15.2	IEEE 802.15 and IEEE 802.11 coexistence	
IEEE 802.15.3	High-Rate wireless PAN	
IEEE 802.15.4	Low-Rate wireless PAN (e.g., ZigBee, WirelessHART, MiWi, etc.)	

IEEE 802.15.5	Mesh networking for WPAN	
IEEE 802.15.6	Body area network )	
IEEE 802.16	Broadband Wireless Access (WiMAX certification)	
IEEE 802.16.1	Local Multipoint Distribution Service	
IEEE 802.17	Resilient packet ring	
IEEE 802.18	Radio Regulatory TAG	
IEEE 802.19	Coexistence TAG	
IEEE 802.20	Mobile Broadband Wireless Access	
IEEE 802.21	Media Independent Handoff	
IEEE 802.22	Wireless Regional Area Network	
IEEE 802.23	Emergency Services Working Group	
IEEE 802.24	Smart Grid TAG	New (November, 2012)
IEEE 802.25	Omni-Range Area Network	Not yet ratified


## References

- IEEE Std 802-1990: IEEE Standards for Local and Metropolitan Networks: Overview and Architecture New York:1990
- [1] "Overview and Guide to the IEEE 802 LMSC" (<http://grouper.ieee.org/groups/802/802/overview.pdf>). September 2004. . Retrieved January 11, 2012. "The project number, 802, was simply the next number in the sequence being issued by the IEEE for standards project"
- [2] "802.3" ([http://www.eeherald.com/section/design-guide/ieee802\\_3.html](http://www.eeherald.com/section/design-guide/ieee802_3.html)). *Data Communication Standards and Protocols*. EE Herald. . Retrieved 2012-01-25.

## External links

- 802 Committee website (<http://www.ieee802.org/>)
- IEEE 802 Standards (<http://standards.ieee.org/getieee802>)

# Ethernet

**Ethernet**  <sup>1</sup> <sup>2</sup> /ˈiːθərnɛt/ is a family of computer networking technologies for local area networks (LANs). Ethernet was commercially introduced in 1980 and standardized in 1985 as IEEE 802.3. Ethernet has largely replaced competing wired LAN technologies.

The Ethernet standards comprise several wiring and signaling variants of the OSI physical layer in use with Ethernet. 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.

Systems communicating over Ethernet divide a stream of data into shorter pieces called frames. Each frame contains source and destination addresses and error-checking data so that damaged data can be detected and re-transmitted. As per the OSI model Ethernet provides services up to and including the data link layer.

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.



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

## History

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

Metcalfe left Xerox in June 1979 to form 3Com.<sup><sup>1</sup><sup>8</sup></sup> 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".<sup><sup>9</sup></sup> Version 2 was published in November, 1982<sup><sup>10</sup></sup> 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.<sup><sup>11</sup></sup> 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.<sup><sup>11</sup><sup>9</sup></sup> 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.<sup><sup>12</sup></sup> An Ethernet adapter card for the IBM PC was released in 1982 and by 1985, 3Com had sold 100,000.<sup><sup>8</sup></sup>

Since then Ethernet technology has evolved to meet new bandwidth and market requirements.<sup><sup>13</sup></sup> In addition to computers, Ethernet is now used to interconnect appliances and other personal devices.<sup><sup>1</sup></sup> It is used in industrial

applications and is quickly replacing legacy data transmission systems in the world's telecommunications networks.<sup>[14]</sup> By 2010, the market for Ethernet equipment amounted to over \$16 billion per year.<sup>[15]</sup>

## Standardization

In February 1980, the Institute of Electrical and Electronics Engineers (IEEE) started project 802 to standardize local area networks (LAN).<sup>[16][8]</sup> 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.<sup>[9]</sup> 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. Competing proposals and broad interest in the initiative led to strong disagreement over which technology to standardize. In December 1980, the group was split into three subgroups, and standardization proceeded separately for each proposal.<sup>[8]</sup>

Delays in the standards process 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.<sup>[11]:8</sup> Because the DIX proposal was most technically complete and because of the speedy action taken by ECMA which decisively contributed to the conciliation of opinions within IEEE, the IEEE 802.3 CSMA/CD standard was approved in December 1982.<sup>[8]</sup> IEEE published the 802.3 standard as a draft in 1983 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 source 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.<sup>[17]</sup> 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.<sup>[18]</sup> Despite the evolution of Ethernet technology, all generations of Ethernet (excluding early experimental versions) use the same frame formats<sup>[19]</sup> (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.<sup>[20]</sup>

## 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,<sup>[21]</sup> 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.<sup>[22]</sup>

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.<sup>[23]</sup> 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.<sup>[24]</sup>

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.<sup>[25]</sup> The network interface card interrupts the CPU only when applicable packets are received: The card ignores information not addressed to it.<sup>[26]</sup> Use of a single cable also means that the bandwidth is shared, such that, for example, available bandwidth to each device is halved when two stations are simultaneously active.

Collisions corrupt transmitted data and require stations to retransmit. The lost data and retransmissions reduce throughput. In the worst case where multiple active hosts connected with maximum allowed cable length attempt to transmit many short frames, excessive collisions can reduce throughput dramatically. However, a Xerox report in 1980 studied performance of an existing Ethernet installation under both normal and artificially generated heavy load. The report claims that 98% throughput on the LAN was observed.<sup>[27]</sup> This is in contrast 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).<sup>[28]</sup>



## 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 more than two 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.<sup>[29]</sup>

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.

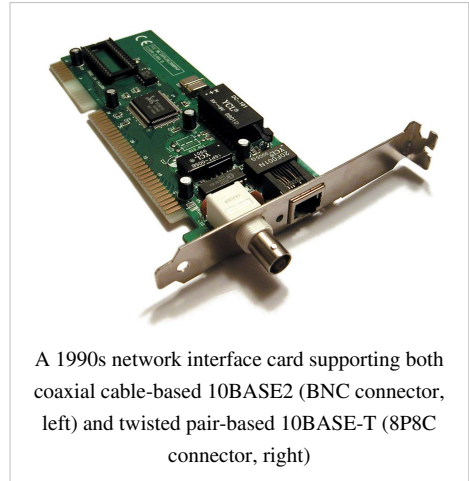
Ethernet on unshielded twisted-pair cables (UTP) began with StarLAN at 1 Mbit/s in the mid-1980s. In 1987 SynOptics introduced the first twisted-pair Ethernet at 10 Mbit/s in a star-wired cabling topology with a central hub, later called LattisNet.<sup>[30][31][8]</sup> These evolved into 10BASE-T, which 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

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 learning of network devices on the different segments, Ethernet bridges (and switches) work somewhat like Ethernet repeaters, passing all traffic between segments. After the bridge learns the addresses associated with each port, it forwards network traffic only to the necessary segments, improving overall



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



Patch cables with patch fields of two Ethernet switches



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 are critical to deployment of Fast Ethernet.

In 1989, the networking company Kalpana introduced their EtherSwitch, the first Ethernet switch.<sup>[32]</sup> This worked somewhat differently from an Ethernet bridge, where 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).<sup>[33]</sup> 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.

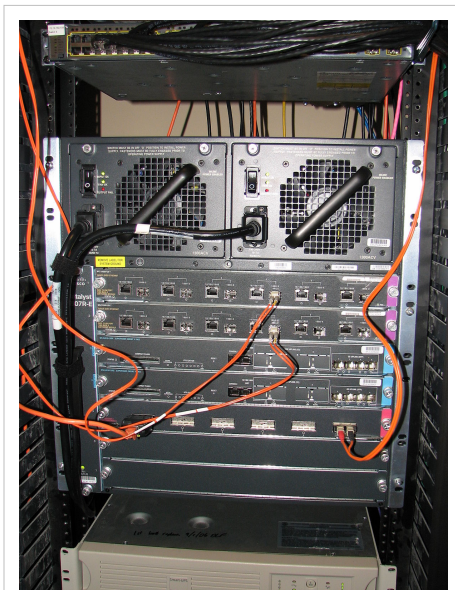
Since packets are typically delivered only to the port they are intended for, traffic on a switched Ethernet is less public than on shared-medium Ethernet. Despite this, switched Ethernet should still be regarded as an insecure network technology, because it is easy to subvert switched Ethernet systems 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.<sup>[34]</sup>

## 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.

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.



A core Ethernet switch

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. In 2012 it was stated by David Allan and Nigel Bragg, in *802.1aq Shortest Path Bridging Design and Evolution: The Architect's Perspective* that Shortest path bridging is one of the most significant enhancements in ethernet's history.<sup>[35]</sup>

## Varieties of Ethernet

The Ethernet physical layer evolved over a considerable time span and encompasses coaxial, twisted pair and fiber optic physical media interfaces and speeds from 10 Mbit to 100 Gbit. 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, e.g. speed and duplex mode. Autonegotiation was an optional feature on first introduction of 100BASE-TX, while 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 0-674-01178-3.
- [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 4,063,220 (<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. This is in contrast to the EtherType in the IEEE Ethernet standard, which specifies the protocol being used.
- [8] Urd Von Burg; Martin Kenny (December 2003). "Sponsors, Communities, and Standards: Ethernet vs. Token Ring in the Local Area Networking Business" ([http://hcd.ucdavis.edu/faculty/webpages/kenney/articles\\_files/Sponsors,Communities,andStandards:Ethernetvs.TokenRingintheLocalAreaNetworkingBusiness.pdf](http://hcd.ucdavis.edu/faculty/webpages/kenney/articles_files/Sponsors,Communities,andStandards:Ethernetvs.TokenRingintheLocalAreaNetworkingBusiness.pdf)). Archived (<http://www.webcitation.org/66LCgXKxh>) from the original on 2012-03-21. .
- [9] 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
- [10] 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
- [11] Robert Breyer & Sean Riley (1999). *Switched, Fast, and Gigabit Ethernet*. Macmillan. ISBN 1-57870-073-6.
  - [12] Jamie Parker Pearson (1992). *Digital at Work*. Digital Press. p. 163. ISBN 1-55558-092-0.
  - [13] 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.
  - [14] "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.
  - [15] 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.
  - [16] 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..."
  - [17] 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.
  - [18] 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.
  - [19] 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."
  - [20] 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."
  - [21] 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.
  - [22] Charles E. Spurgeon (2000). *Ethernet: The Definitive Guide*. O'Reilly. ISBN 978-1-56592-660-8.
  - [23] 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.
  - [24] 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.
  - [25] 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.
  - [26] Unless it is put into promiscuous mode.
  - [27] 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. .
  - [28] 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. .
  - [29] 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.
  - [30] 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](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. .
  - [31] 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. .
  - [32] The term *switch* was invented by device manufacturers and does not appear in the 802.3 standard.
  - [33] This is misleading, as performance will double only if traffic patterns are symmetrical.
  - [34] "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](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"
  - [35] 802.1aq Shortest Path Bridging Design and Evolution: The Architect's Perspective ([http://www.eabooks.com.au/epages/eab.sf/en\\_au/?ObjectPath=/Shops/eabooks/Products/9781118148662](http://www.eabooks.com.au/epages/eab.sf/en_au/?ObjectPath=/Shops/eabooks/Products/9781118148662))

## 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](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. Data on Ethernet is transmitted most-significant byte first. Within each byte, however, the least-significant bit is transmitted first.<sup>[1]</sup>

The table below shows the complete Ethernet frame, as transmitted, for the MTU of 1500 octets.<sup>[2]</sup> Some implementations of Gigabit Ethernet (and higher speeds) support larger jumbo frames.

#### 802.3 Ethernet frame structure

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42 <sup>[3]</sup> –1500 octets	4 octets	12 octets
		64–1522 octets						
		72–1530 octets						
		84–1542 octets						

## Preamble and start frame delimiter

A frame starts with a 7-octet preamble and 1-octet start frame delimiter (SFD).<sup>[4]</sup> Prior to Fast Ethernet, the on-the-wire bit pattern for this portion of the frame is 10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011.<sup>[5]</sup> Since octets are transmitted least-significant bit first the corresponding hexadecimal representation is 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0xD5.

PHY transceiver chips used for Fast Ethernet feature a 4-bit (one nibble) Media Independent Interface. Therefore the preamble will consist of 14 instances of 0x5, and the start frame delimiter 0x5 0xD. Gigabit Ethernet transceiver chips use a Gigabit Media Independent Interface that works 8-bits at a time, and 10 Gbit/s (XGMII) PHY works with 32-bits at a time. by Narayanan

## Header

The header features source and destination MAC addresses which have 6 octets each, the EtherType protocol identifier field and optional IEEE 802.1Q tag.

### 802.1Q tag

The IEEE 802.1Q tag is an optional 4-octet field that indicates Virtual LAN (VLAN) membership and IEEE 802.1p priority.

### EtherType or length

EtherType is a two-octet field in an Ethernet frame. It is used to indicate which protocol is encapsulated in the payload of an Ethernet Frame.

## Payload

The minimum payload is 42 octets when 802.1Q tag is present and 46 octets when absent.<sup>[6]</sup> and the maximum payload is 1500 octets. Non-standard jumbo frames allow for larger maximum payload size.

## Frame check sequence

The frame check sequence is a 4-octet cyclic redundancy check which allows detection of corrupted data within the entire frame.

## Interframe gap

Interframe gap is idle time between frames. 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<sup>[7]</sup> 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 an IEEE 802.1Q tag to identify what VLAN it belongs to and its priority (quality of service). This encapsulation is defined in the IEEE 802.3ac specification and increases the maximum frame by 4 octets.

---

The IEEE 802.1Q tag, if present, is placed between the Source Address and the EtherType or Length fields. The first two octets 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 octets 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<sup>[8]</sup>), 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.<sup>[9]</sup> 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 payload field of an Ethernet 802.3 frame is 1500 octets (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.<sup>[10]</sup> 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.<sup>[11]</sup> 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.<sup>[12]</sup> 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 octets (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.<sup>[13]</sup>

## Maximum throughput

We may calculate the protocol overhead for Ethernet as a percentage

$$\text{Protocol overhead} = \frac{\text{Frame size} - \text{Payload size}}{\text{Frame size}}$$

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 a 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 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 octets. Possible causes are collision, underruns, a bad network card or software.<sup>[14]</sup>

## Notes

- [1] "Ethernet Frame" (<http://www.infocellar.com/networks/ethernet/frame.htm>). . Retrieved 2012-03-20. "Ethernet transmission is strange, in that the byte order is big-endian (leftmost byte is sent first), but bit order little-endian (rightmost, or LSB (Least Significant Bit) of the byte is sent first)."
- [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] 42 octet minimum applies when 802.1Q is present. When absent, 46 octet minimum applies. IEEE 802.3-2005 Clause 3.5
- [4] Preamble and start frame delimiter are not displayed by packet sniffing software because these bits are stripped away at OSI Layer 1 by the network interface controller before being passed on to the OSI layer 2 which is where packet sniffers collect their data. There are layer-2 sniffers which can capture and display the preamble and start frame delimiter but they are expensive and mainly used to detect physical related problems.
- [5] IEEE 802.3 Clause 4.2.5
- [6] Minimum payload size is dictated by the slot time used for collision detection in the Ethernet LAN architecture.
- [7] A version 1 Ethernet frame was used for early Ethernet prototypes and featured 8-bit MAC addresses and was never commercially deployed.
- [8] Drew Heywood; Zubair Ahmad (2001). *Drew Heywood's Windows 2000 Network Services*. Sams. p. 53. ISBN 0-672-31741-9.
- [9] Original Ethernet packets define their length with the framing that surrounds it, rather than with an explicit length count.
- [10] 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.
- [11] IEEE Std 802.3-2005, 3.2.6
- [12] "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. .
- [13] 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](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.
- [14] "Troubleshooting Ethernet" (<http://www.cisco.com/en/US/docs/internetworking/troubleshooting/guide/tr1904.html>). Cisco Systems.

## References



# 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. As of 2012 IPv4 is still the most widely deployed Internet Layer protocol.

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. Addresses were assigned to users, and the number of unassigned addresses decreased. IPv4 address exhaustion occurred on February 3, 2011. It had been significantly delayed by address changes such as classful network design, Classless Inter-Domain Routing, and network address translation (NAT).

This limitation of IPv4 stimulated the development of IPv6 in the 1990s, which has been in commercial deployment since 2006.

IPv4 reserves special address blocks for private networks (~18 million addresses) and multicast addresses (~270 million addresses).

## 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 the 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
Dotted decimal	192.0.2.235	N/A
Dotted hexadecimal <sup>[1]</sup>	0xC0.0x00.0x02.0xEB	Each octet is individually converted to hexadecimal form
Dotted octal <sup>[1]</sup>	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

## Allocation

Originally, an IP address was divided into two parts: the network identifier was the most significant (highest order) octet of the address, and the host identifier was 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, people devised methods to subdivide IP networks. One flexible method was the *variable-length subnet mask* (VLSM).<sup>[2][3]</sup>

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

Range	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
100.64.0.0/10	Shared Address Space	RFC 6598
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	IETF Protocol Assignments	RFC 5735
192.0.2.0/24	TEST-NET-1, documentation and examples	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	IP multicast (former Class D network)	RFC 5771
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. Two private networks (e.g., two branch offices) cannot communicate via the public internet, unless they use an IP tunnel or a virtual private network (VPN). When one private network wants to send a packet to another private network, the first private network encapsulates the packet in a protocol layer so that the packet can travel through the public network. Then the packet travels through the public network. When the packet reaches the other private network, its protocol layer is removed, and the packet travels to its destination.

Optionally, encapsulated packets may be encrypted to secure the data while it travels over the public network.

## Link-local addressing

RFC 5735 defines the special address block 169.254.0.0/16 for link-local addressing. These addresses are only valid on links (such as a local network segment or point-to-point connection) connected to a host. These addresses are not routable. Like private addresses, these addresses cannot be the source or destination of packets traversing the internet. These 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 address autoconfiguration. Microsoft created an implementation called Automatic Private IP Addressing (APIPA), which was deployed on millions of machines and became a de facto standard. Many years later, in May 2005, the IETF defined a formal standard in RFC 3927, entitled *Dynamic Configuration of IPv4 Link-Local Addresses*.

## Loopback

The class A network 127.0.0.0 (classless network 127.0.0.0/8) is reserved for loopback. IP packets which source addresses belong to this network should never appear outside a host. The modus operandi of this network expands upon that of a loopback interface:

- IP packets which source and destination addresses belong to the network (or subnetwork) of the same loopback interface are returned back to that interface;
- IP packets which source and destination addresses belong to networks (or subnetworks) of different interfaces of the same host, one of them being a loopback interface, are forwarded regularly.

## 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, consider a /16 subnet 192.168.0.0/255.255.0.0, which is equivalent to the address range 192.168.0.0–192.168.255.255. The broadcast address is 192.168.255.255. One can use the following addresses for hosts, even though they end with 255: 192.168.1.255, 192.168.2.255, etc. Also, 192.168.0.0 is the network identifier and must not be used for a host.<sup>[4]</sup> One can use the following addresses for hosts, even though they end with 0: 192.168.1.0, 192.168.2.0, etc.

In the past, conflict between network addresses and broadcast addresses arose because some software used non-standard broadcast addresses with zeros instead of ones.<sup>[5]</sup>

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. This is analogous to looking up a phone number in a phone book using the recipient's name.

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 being depleted at a rate that was not initially anticipated in the original design of the network address system.<sup>[6]</sup> The threat of exhaustion was the motivation for remedial technologies, such as classful networks, Classless Inter-Domain Routing (CIDR) methods, and network address translation (NAT). Eventually, IPv6 was created, which has many more addresses available.

Several market forces accelerated IPv4 address exhaustion:

- Rapidly growing number of Internet users
- Always-on devices — ADSL modems, cable modems
- Mobile devices — laptop computers, PDAs, mobile phones

Some technologies mitigated IPv4 address exhaustion:

---

- Network address translation (NAT) is a technology that allows a private network to use one public IP address. It permits private addresses in 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.<sup>[7][8]</sup> 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.<sup>[9]</sup>

The accepted and standard solution is to use 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.

#### IPv4 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				IHL				DSCP				ECN				Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

#### 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), which is 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  $5 \times 32 = 160$  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), which is used for interactive data voice exchange.

#### Explicit Congestion Notification (ECN)

This field is 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 packet (fragment) size, including header and data, in bytes. The minimum-length packet 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 largest datagram that any host is required to be able to reassemble is 576 bytes, but most modern hosts handle much larger packets. Sometimes subnetworks impose further restrictions on the packet size, in which case datagrams must be fragmented. Fragmentation is handled in either the host or router 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 help trace datagrams with spoofed source addresses.<sup>[10]</sup>

#### 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.<sup>[11]</sup>
- 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 practice, the field has become a hop count—when the datagram arrives at a router, the router decrements the TTL field by one. When the TTL field hits zero, the router discards the packet and typically sends a ICMP Time Exceeded message to the sender.

---

The program traceroute uses these ICMP Time Exceeded messages to print the routers used by packets to go from the source to the destination.

#### 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. When a packet arrives at a router, the router calculates the checksum of the header and compares it to the checksum field. If the values do not match, the router discards the packet. Errors in the data field must be handled by the encapsulated protocol. Both UDP and TCP have checksum fields.

When a packet arrives at a router, the router decreases the TTL field. Consequently, the router must calculate a new checksum. RFC 1071 defines the checksum calculation:

*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, consider Hex 4500003044224000800600008c7c19acae241e2b (20 bytes IP header):

Step 1)  $4500 + 0030 + 4422 + 4000 + 8006 + 0000 + 8c7c + 19ac + ae24 + 1e2b = 2BBCF$  (16-bit sum)

Step 2)  $2 + BBCF = BBD1 = 1011101111010001$  (1's complement 16-bit sum)

Step 3)  $\sim BBD1 = 0100010000101110 = 442E$  (1's complement of 1's complement 16-bit sum)

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

This field is the IPv4 address of the sender of the packet. Note that this address may be changed in transit by a network address translation device.

#### Destination address

This field is the IPv4 address of the receiver of the packet. As with the source address, this may be changed in transit by a network address translation device.

#### Options

The options field is 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 following two options are discouraged because they create security concerns: Loose Source and Record Route (LSRR) and Strict Source and Record Route (SSRR). Many routers block packets containing these options.<sup>[12]</sup>

## 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 enables networks to communicate with one another. The design accommodates networks of diverse physical nature; it is independent of the underlying transmission technology used in the Link Layer. Networks with different hardware usually vary not only in transmission speed, but also in the maximum transmission unit (MTU). When one network wants to transmit datagrams to a network with a smaller MTU, it may fragment its datagrams. In IPv4, this function was placed at the Internet Layer, and is performed in IPv4 routers, which thus only



require this layer as the highest one implemented in their design.

In contrast, IPv6, the next generation of the Internet Protocol, does not allow routers to perform fragmentation; hosts must determine the path MTU before sending datagrams.

## Fragmentation

When a router receives a packet, it examines the destination address and determines the outgoing interface to use. The interface has an MTU. If the packet size is bigger than the MTU, the router may fragment the packet.

The router divides the packet into segments. The max size of each segment is the MTU minus the IP header size (20 bytes minimum; 60 bytes maximum). The router puts each segment into its own packet, each fragment packet having following changes:

- The *total length* field is 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, 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 MTU of 1,500 bytes and a header size of 20 bytes, the fragment offsets would be multiples of  $(1500 - 20)/8 = 185$ . These multiples are 0, 185, 370, 555, 740, ...

It is possible for a packet to be fragmented at one router, and for the fragments to be fragmented at another router. For example, consider a packet with a data size of 4,500 bytes, no options, and a header size of 20 bytes. So the packet size is 4,520 bytes. Assume that the packet travels over a link with an MTU of 2,500 bytes. Then it will become two fragments:

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (bytes)
1	2500	20	2480	1	0
2	2040	20	2020	0	310

Note that the fragments preserve the data size:  $2480 + 2020 = 4500$ .

Note how we get the offsets from the data sizes:

- 0.
- $0 + 2480/8 = 310$ .

Assume that these fragments reach a link with an MTU of 1,500 bytes. Each fragment will become two fragments:

Fragment	Total bytes	Header bytes	Data bytes	"More fragments" flag	Fragment offset (bytes)
1	1500	20	1480	1	0
2	1020	20	1000	1	185
3	1500	20	1480	1	310
4	560	20	540	0	495

Note that the fragments preserve the data size:  $1480 + 1000 = 2480$ , and  $1480 + 540 = 2020$ .

Note how we get the offsets from the data sizes:

- 0.
- $0 + 1480/8 = 185$
- $185 + 1000/8 = 310$
- $310 + 1480/8 = 495$

We can use the last offset and last data size to calculate the total data size:  $495*8 + 540 = 3960 + 540 = 4500$ .

## Reassembly

A receiver knows that a packet is a fragment if at least one of the following conditions is true:

- The "more fragments" flag is set. (This is true for all fragments except the last.)
- The "fragment offset" field is nonzero. (This is true for all fragments except the first.)

The receiver identifies matching fragments using the identification field. The receiver will reassemble the data from fragments with the same identification field using both the fragment offset and the more fragments flag. When the receiver receives the last fragment (which has the "more fragments" flag set to 0), it can calculate the length of the original data payload, by multiplying the last fragment's offset by eight, and adding the last fragment's data size. In the example above, this calculation was  $495 \times 8 + 540 = 4500$  bytes.

When the receiver has all the fragments, it can put them in the correct order, by using their offsets. It can then pass their data 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 translation for IPv4. (A hardware address is also called a MAC address.) 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/](http://www.unix.com/man-page/Linux/3/inet_addr/)). . Retrieved 2010-11-28.
- [2] "Planning Classless Routing: TCP/IP" ([http://technet.microsoft.com/en-us/library/cc779089\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc779089(WS.10).aspx)). Technet.microsoft.com. 2003-03-28. . Retrieved 2012-01-20.
- [3] "HP Networking: switches, routers, wired, wireless, HP TippingPoint Security" ([http://www.3com.com/other/pdfs/infra/corpinfo/en\\_US/501302.pdf](http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf)). 3com.com. . Retrieved 2012-01-20.
- [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".
- [12] "Cisco unofficial FAQ" (<http://www.faqs.org/faqs/cisco-networking-faq/section-23.html>). . Retrieved 2012-05-10.

## 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](http://www.inetcore.com/project/ipv4ec/index_en.html)) (estimated)

## 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 with a minimum of protocol mechanism.<sup>[1]</sup> It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. As this is normally IP over unreliable media, there is no guarantee of delivery, ordering or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

UDP is suitable for purposes where 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.<sup>[2]</sup> 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.

A number of UDP's attributes make it especially suited for certain applications.

- It is **transaction-oriented**, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
  - It provides **datagrams**, suitable for modeling other protocols such as in IP tunneling or Remote Procedure Call and the Network File System.
  - It is **simple**, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
-

- It is **stateless**, suitable for very large numbers of clients, such as in streaming media applications for example IPTV
- The **lack of retransmission delays** makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Works well in **unidirectional** communication, suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol

## 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.<sup>[3]</sup> 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.<sup>[3]</sup> 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.<sup>[4]</sup>

UDP provides application multiplexing (via port numbers) and integrity verification (via checksum) of the header and payload.<sup>[5]</sup> If transmission reliability is desired, it must be implemented in the user's application.

Offset (bits)	Field
0	Source Port Number
16	Destination Port Number
32	Length
48	Checksum
64+	Data □

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits).<sup>[2]</sup> 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.<sup>[3]</sup>

### 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.<sup>[3]</sup>

#### 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).<sup>[3]</sup>

In IPv6 Jumbograms it is possible to have UDP packets of size greater than 65,535 bytes.<sup>[6]</sup> This allows for a maximum length value of 4,294,967,295 bytes ( $2^{32} - 1$ ) with 8 bytes representing the header and 4,294,967,287 bytes for data.

#### 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.<sup>[7]</sup> This field is not optional for IPv6.<sup>[8]</sup>

## 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.*<sup>[7]</sup>

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.

## Pseudo-Headers

### IPv4 Pseudo Header

When UDP runs over IPv4, the checksum is computed using a "pseudo header"<sup>[9]</sup> 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, it 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.<sup>[8]</sup>*

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.<sup>[3]</sup>

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 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)<sup>[2]</sup> 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.<sup>[10]</sup>

## 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] RFC 768 p1
- [2] Kurose, J. F.; Ross, K. W. (2010). *Computer Networking: A Top-Down Approach* (5th ed.). Boston, MA: Pearson Education. ISBN 978-0-13-136548-3.
- [3] Forouzan, B.A. (2000). *TCP/IP: Protocol Suite, 1st ed.* New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
- [4] content@ipv6.com. "UDP Protocol Overview" (<http://archive.is/20120710/http://ipv6.com/articles/general/User-Datagram-Protocol.htm>). Ipv6.com. Archived from the original (<http://ipv6.com/articles/general/User-Datagram-Protocol.htm>) on 2012-07-10. . Retrieved 17 August 2011.
- [5] Clark, M.P. (2003). *Data Networks IP and the Internet, 1st ed.* West Sussex, England: John Wiley & Sons Ltd.
- [6] RFC 2675
- [7] "Postel, J. (August 1980). RFC 768: User Datagram Protocol. *Internet Engineering Task Force*. Retrieved from" (<http://archive.is/20120722/http://tools.ietf.org/html/rfc768>). Archived from the original (<http://tools.ietf.org/html/rfc768>) on 2012-07-22. .
- [8] "Deering S. & Hinden R. (December 1998). RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. *Internet Engineering Task Force*. Retrieved from" (<http://archive.is/20120915/http://tools.ietf.org/html/rfc2460>). Archived from the original (<http://tools.ietf.org/html/rfc2460>) on 2012-09-15. .
- [9] RFC 768, p2
- [10] "The impact of UDP on Data Applications" ([http://www.networkperformancedaily.com/2007/08/whiteboard\\_series\\_nice\\_guys\\_fi.html](http://www.networkperformancedaily.com/2007/08/whiteboard_series_nice_guys_fi.html)). Networkperformancedaily.com. . Retrieved 17 August 2011.

## RFC references

- RFC 768 – User Datagram Protocol
- RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification
- RFC 2675 - IPv6 Jumbograms
- 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>)
-



---

# 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 octets from a program on one computer to another program on another computer. TCP is the protocol used by major Internet applications such as the World Wide Web, email, remote administration and file transfer. 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 Intercommunication*."<sup>[1]</sup> 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

The protocol corresponds to the transport layer of TCP/IP suite. 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.<sup>[2]</sup>

TCP is a reliable stream delivery service that guarantees that all bytes received will be identical with bytes sent and in the correct order. 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. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires before the message has been acknowledged. The timer is needed in case a packet gets lost or corrupted.<sup>[2]</sup>

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."<sup>[3]</sup>

The term *TCP packet*, though sometimes informally used, is not in line with current terminology, where *segment* refers to the TCP PDU, *datagram*<sup>[4]</sup> 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.<sup>[5]</sup>

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*, orange 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 0 0 0			N	C	E	U	A	P	R	S	F	Window Size																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										

- 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 sender of this segment 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.<sup>[6]</sup>

Some options may only be sent when SYN is set; they are indicated below as <sup>[SYN]</sup>. 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*) <sup>[SYN]</sup>
- 3,3,S (24 bits) – Window scale (*see window scaling for details*) <sup>[SYN]</sup><sup>[7]</sup>
- 4,2 (16 bits) – Selective Acknowledgement permitted. <sup>[SYN]</sup> (*See selective acknowledgments for details*)<sup>[8]</sup>
- 5,N,BBBB,EEEE,... (variable bits, N is either 10, 18, 26, or 34)- Selective ACKnowledgement (SACK)<sup>[9]</sup>  
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*)<sup>[10]</sup>
- 14,3,S (24 bits) – TCP Alternate Checksum Request. <sup>[SYN]</sup><sup>[11]</sup>
- 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 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 the local end-point undergoes a series of state changes:<sup>[13]</sup>

### LISTEN

(server) represents waiting for a connection request from any remote TCP and port.

### SYN-SENT

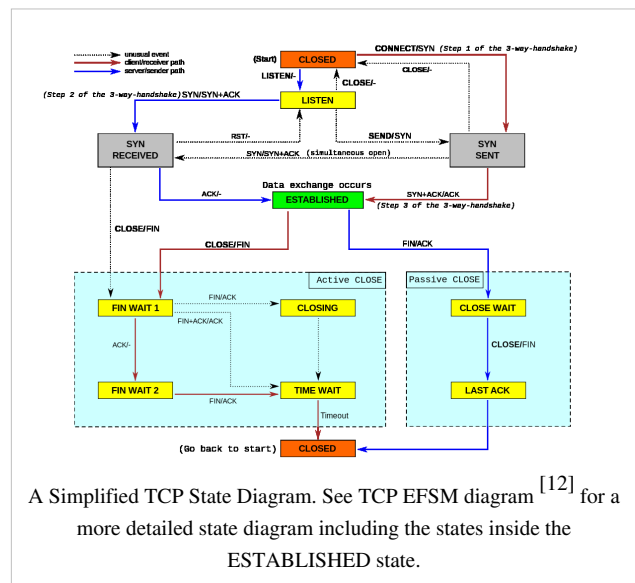
(client) represents waiting for a matching connection request after having sent a connection request.

### SYN-RECEIVED

(server) represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

### ESTABLISHED

(both server and client) represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.



**FIN-WAIT-1**

(both server and client) represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

**FIN-WAIT-2**

(both server and client) represents waiting for a connection termination request from the remote TCP.

**CLOSE-WAIT**

(both server and client) represents waiting for a connection termination request from the local user.

**CLOSING**

(both server and client) represents waiting for a connection termination request acknowledgment from the remote TCP.

**LAST-ACK**

(both server and client) represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

**TIME-WAIT**

(either server or client) represents waiting for enough time to pass to be sure the remote TCP 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).]

**CLOSED**

(both server and client) represents no connection state at all.

**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 and listen at 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:

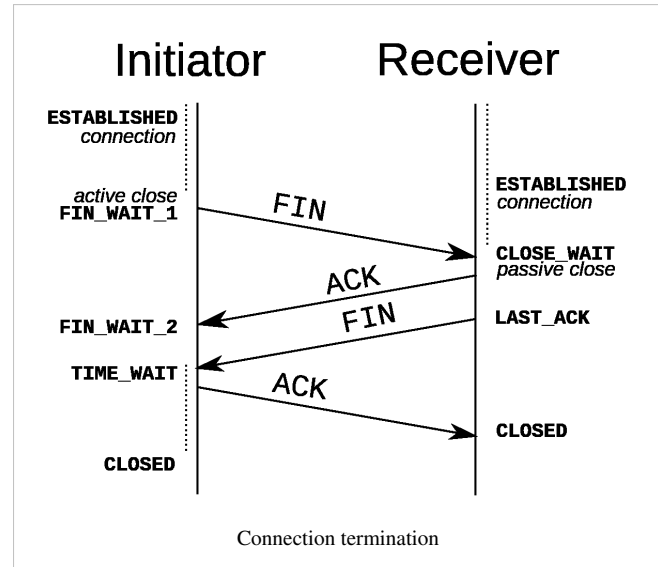
1. **SYN:** The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
2. **SYN-ACK:** In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number ( $A + 1$ ), 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. The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

---

## Connection termination

The connection termination phase uses 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 side which sent the first FIN before receiving one 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.<sup>[14]</sup> 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<sup>[15]</sup>). 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.

## 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<sup>[2]</sup>
- Retransmission of lost packets — any cumulative stream not acknowledged is retransmitted<sup>[2]</sup>
- Error-free data transfer<sup>[16]</sup>
- 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.<sup>[2]</sup>
- Congestion control<sup>[2]</sup>

## 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.<sup>[17]</sup> This is the end-to-end principle at work.

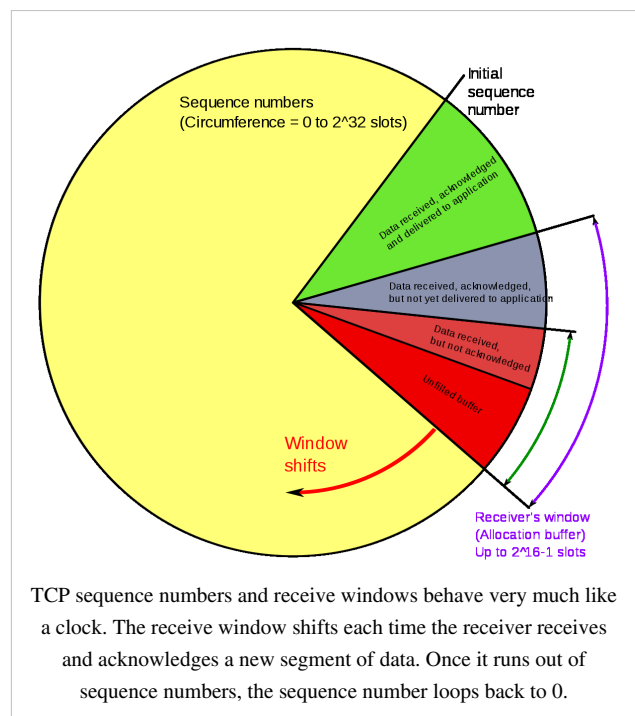
## Flow control

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 smartphone that is slowly processing received data, the smartphone must regulate the data flow so as not to be overwhelmed.<sup>[2]</sup>

TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the *receive window* field the amount of additionally 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.

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 6298. 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.<sup>[18]</sup> This situation may arise, for example, if one of the devices participating in a connection has an extremely limited amount of 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.<sup>[19]</sup>

## 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 ticks 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 4 GB 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 1 GB (which is correct, but the sequence number limit is 4 GB); 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.<sup>[2]</sup>

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 is reliable at all as it is one of the least commonly used protocol elements and tends to be poorly implemented.<sup>[20][21]</sup>

## 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 4 KB, a typical MSS is 1460, so 2 packets go out on a 10 Mbit/s ethernet taking ~1.2 ms each followed by a third carrying the remaining 1176 after a 197 ms 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 200 ms 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".<sup>[2]</sup> There is no way to indicate or control it in User space using Berkeley sockets and it is controlled by Protocol stack only.<sup>[22]</sup>

## 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,<sup>[23]</sup> and is currently being pursued within the IETF.<sup>[24]</sup>

## 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.<sup>[25]</sup> An advanced DoS attack involving the exploitation of the TCP Persist Timer was analyzed at Phrack #66.<sup>[26]</sup>

## 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.<sup>[27]</sup>

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 now 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), SSL (443) 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)<sup>[28]</sup> 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)<sup>[29][30]</sup> 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<sup>[31]</sup> enables statistical multiplexing of resources, and thus increases TCP throughput

dramatically. Multipath TCP also brings performance benefits in datacenter environments.<sup>[32]</sup> The reference implementation<sup>[33]</sup> of Multipath TCP is being developed in the Linux kernel.<sup>[34]</sup>

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 is 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 Fast Open is an extension to speed up the opening of successive TCP connections between two endpoints. It works by skipping the three-way handshake using a cryptographic "cookie". It is similar to an earlier proposal called T/TCP, which was not widely adopted due to security issues.<sup>[35]</sup> As of July 2012, it is an IETF Internet draft.<sup>[36]</sup>

## 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),<sup>[37]</sup> link layer solutions (such as RLP in cellular networks), or proxy based solutions (which require some changes in the network without modifying end nodes).<sup>[37][38]</sup>

A number of alternative congestion control algorithms have been proposed to help solve the wireless problem, such as Vegas, Westwood, Veno and Santa Cruz.

## 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 problem (at least with normal implementations) is that the application cannot access 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 generally 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 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 the OS from using precious CPU cycles calculating the 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, (May 1974). "A Protocol for Packet Network Intercommunication" (<http://ece.ut.ac.ir/Classpages/F84/PrincipleofNetworkDesign/Papers/CK74.pdf>). *IEEE Transactions on Communications* **22** (5): 637–648. .
- [2] Comer, Douglas E. (2006). *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. **1** (5th ed.). Prentice Hall. ISBN 0-13-187671-6.
- [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] Tanenbaum, Andrew S. (2003-03-17). *Computer Networks* (Fourth ed.). Prentice Hall. ISBN 0-13-066102-3.
- [15] <http://tools.ietf.org/html/rfc1122>
- [16] "TCP Definition" (<http://www.linfo.org/tcp.html>). . Retrieved 2011-03-12.
- [17] Stone; Partridge (2000). "When The CRC and TCP Checksum Disagree" (<http://citeseer.ist.psu.edu/stone00when.html>). *Sigcomm*.
- [18] RFC 879 (<http://www.faqs.org/rfcs/rfc879.html>)
- [19] TCP window scaling and broken routers [LWN.net] (<http://lwn.net/Articles/92727/>)
- [20] 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.
- [21] Peterson, Larry (2003). *Computer Networks*. Morgan Kaufmann. p. 401. ISBN 1-55860-832-X.
- [22] 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.
- [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] Some insights about the recent TCP DoS (Denial of Service) vulnerabilities (<http://www.gont.com.ar/talks/hacklu2009/fgont-hacklu2009-tcp-security.pdf>)
- [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](http://www.medianet.kent.edu/projects_files/projectITCP.html))
- [29] RFC 6182
- [30] "TCP Extensions for Multipath Operation with Multiple Addresses draft-ietf-mptcp-multiaddressed-09" (<http://tools.ietf.org/html/draft-ietf-mptcp-multiaddressed-09>). [tools.ietf.org](http://tools.ietf.org/html/draft-ietf-mptcp-multiaddressed-09). Archived from the original (<http://datatracker.ietf.org/doc/draft-ietf-mptcp-multiaddressed/>) on June 6, 2012. .
- [31] TCP with feed-forward source coding for wireless downlink networks (<http://portal.acm.org/citation.cfm?id=1794199>)
- [32] Raiciu; Barre; Plunke; 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] MultiPath TCP - Linux Kernel implementation (<http://mptcp.info.ucl.ac.be/>)
- [34] Barre; Paasch; Bonaventure (2011). "MultiPath TCP: From Theory to Practice" (<http://inl.info.ucl.ac.be/publications/multipath-tcp-theory-practice>). *IFIP Networking*.
- [35] Michael Kerrisk (2012-08-01). "TCP Fast Open: expediting web services" (<https://lwn.net/SubscriberLink/508865/39212876277c174c/>). LWN.net. .
- [36] Y. Cheng, J. Chu, S. Radhakrishnan, A. Jain (2012-07-16). *TCP Fast Open* (<https://tools.ietf.org/html/draft-ietf-tcpm-fastopen-01>). IETF. I-D draft-ietf-tcpm-fastopen-01. .
- [37] "TCP performance over CDMA2000 RLP" (<http://academic.research.microsoft.com/Paper/3352358.aspx>). . Retrieved 2010-08-30
- [38] 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 4614 – A Roadmap for TCP Specification Documents
- RFC 5681 – TCP Congestion Control
- RFC 6298 – Computing TCP's Retransmission Timer

### Others

- Oral history interview with Robert E. Kahn (<http://purl.umn.edu/107387>), 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\\_history.php?simu=tcp\\_fast\\_retransmit&protocol=TCP&title=4.Fast transmit&ctype=1](http://www.visualland.net/tcp_history.php?simu=tcp_fast_retransmit&protocol=TCP&title=4.Fast%20transmit&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](http://www.linktionary.com/s/segment_tcp.html))
- TCP Sliding Window simulation animated (ns2) ([http://www.visualland.net/tcp\\_history.php?simu=tcp\\_swnd&protocol=TCP&title=2.Sliding Window&ctype=1](http://www.visualland.net/tcp_history.php?simu=tcp_swnd&protocol=TCP&title=2.Sliding%20Window&ctype=1))
- Multipath TCP in Linux kernel (<http://inl.info.ucl.ac.be/mptcp>)

## OSI model

The **Open Systems Interconnection (OSI) model** (ISO/IEC 7498-1) 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. A layer serves the layer above it and is served by the layer below it.

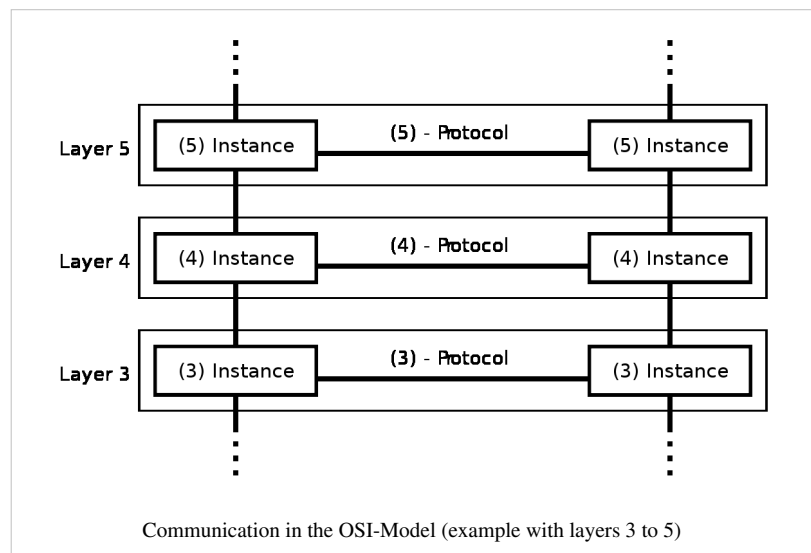
For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of that path. Two instances at one layer are connected by a horizontal connection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

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.<sup>[1]</sup> 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.<sup>[2]</sup>

## Description of OSI layers

According to recommendation X.200, there are seven layers, labeled 1 to 7, with layer 1 at the bottom. Each layer is generically known as an N layer. An "N+1 entity" (at layer N+1) requests services from an "N entity" (at layer N).

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 a layer N is the SDU of layer N-1. In effect the SDU is the 'payload' of a given PDU. That is, the process of changing an 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. 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
<b>Host layers</b>	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
<b>Media layers</b>	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.<sup>[3]</sup>

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 fiber optical cable. This includes the layout of pins, voltages, line impedance, cable specifications, signal timing, 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 multi-access media, was developed independently of the ISO work in IEEE Project 802. IEEE work assumed sublayer-ing 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 multi-access media.

While IEEE 802.3 is the dominant wired LAN protocol and IEEE 802.11 the wireless LAN protocol, obsolete 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 (in contrast to the data link layer which connects hosts within the same network), while maintaining the quality of service requested by the transport layer. 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 IPv6 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.<sup>[4]</sup>

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

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. On this level, Inter-Process communication happen (SIGHUP, SIGKILL, End Process, etc.).

## 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.<sup>[5]</sup>

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 include the following:

- security service (telecommunication)<sup>[3]</sup> 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.
- Multiprotocol Label Switching (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, known as network sockets 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 <sup>[6]</sup>	AppleTalk	IPX	SNA	UMTS	Misc. examples
#	Name								
7	Application	FTAM, X.400, X.500, DAP, ROSE, RTSE, ACSE <sup>[7]</sup> , CMIP <sup>[8]</sup>	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, SOCKS
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) and BMC (Broadcast/Multicast Control)	NBF, Q.931, NDP ARP (maps layer 3 to layer 2 address), IS-IS



2	Data Link	ISO/IEC 7666, X.25 (LAPB), Token Bus, X.222, ISO/IEC 8802-2 LLC Type 1 and 2 <sup>[9]</sup>	PPP, SBT, SLIP, PPTP	MTP, Q.710	LocalTalk, AppleTalk Remote Access, PPP	IEEE 802.3 framing, Ethernet II framing	SDLC	Packet Data Convergence Protocol (PDCP) <sup>[10]</sup> , LLC (Logical Link Control), MAC (Media Access Control)	802.3 (Ethernet), 802.11a/b/g/n MAC/LLC, 802.1Q (VLAN), ATM, HDLC, FDDI, Fibre Channel, Frame Relay, HDLC, ISL, PPP, Q.921, Token Ring, CDP, 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) <sup>[9]</sup>		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, 1000BASE-T, 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 model

In the TCP/IP model of the Internet, protocols are deliberately not as rigidly designed into strict layers as in the OSI model.<sup>[11]</sup> RFC 3439 contains a section entitled "Layering considered harmful (section link here [12])." 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 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] 3GPP TS 36.300 : E-UTRA and E-UTRAN Overall Description, Stage 2, Release 11 (<http://www.3gpp.org/ftp/Specs/html-info/36300.htm>)
- [11] RFC 3439
- [12] <http://tools.ietf.org/html/rfc3439#section-3>

## External links

- ISO/IEC standard 7498-1:1994 ([http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269\\_ISO\\_IEC\\_7498-1\\_1994\(E\).zip](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](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>)
- Zimmermann, Hubert (April 1980). "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection". *IEEE Transactions on Communications* **28** (4): 425–432. CiteSeerX: 10.1.1.136.9497 (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.136.9497>).
- Cisco Systems Internetworking Technology Handbook ([http://docwiki.cisco.com/wiki/Internetworking\\_Technology\\_Handbook](http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook))
- Collection of animations and videos concerning computer networks (<http://www.khurramtanvir.com/cs460demos.php>)

# Internet protocol suite

The **Internet protocol suite** is the set of communications protocols used for the Internet and similar networks, and generally the most popular protocol stack for wide area networks. It is commonly known as **TCP/IP**, because of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP), which were the first networking protocols defined in this standard. It is occasionally known as the **DoD model** due to the foundational influence of the ARPANET in the 1970s (operated by DARPA, an agency of the United States Department of Defense).

TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. It has four abstraction layers, each with its own protocols.<sup>[1][2]</sup> From lowest to highest, the layers are:

1. The link layer (commonly Ethernet) contains communication technologies for a local network.
2. The internet layer (IP) connects local networks, thus establishing internetworking.
3. The transport layer (TCP) handles host-to-host communication.
4. The application layer (for example HTTP) contains all protocols for specific data communications services on a process-to-process level (for example how a web browser communicates with a web server).

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

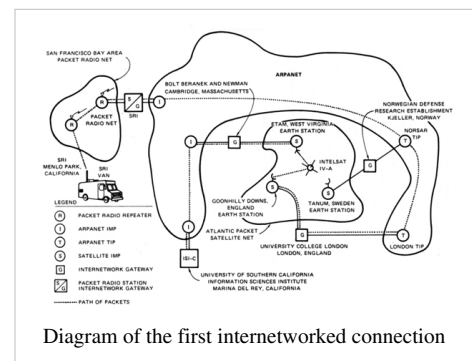
## History

### Early research

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.

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



A Stanford Research Institute packet radio van, site of the first three-way internetworked transmission.

eventual product of Cerf and Kahn's work, will run over "*two tin cans and a string*." As a joke, the IP over Avian Carriers formal protocol specification was created and successfully tested.

A computer called a router is provided with an interface to each network. It forwards packets back and forth between them.<sup>[3]</sup> Originally a router was called *gateway*, but the term was changed to avoid confusion with other types of gateways.

## Specification

From 1973 to 1974, Cerf's networking research group at Stanford worked out details of the idea, resulting in the first TCP specification.<sup>[4]</sup> A significant technical influence was the early networking work at Xerox PARC, which produced the PARC Universal Packet protocol suite, much of which existed around that time.

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, TCP v3 and IP v3, and TCP/IP v4. The last protocol is still in use 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, the 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.<sup>[5]</sup>

## Adoption

In March 1982, the US Department of Defense declared TCP/IP as the standard for all military computer networking.<sup>[6]</sup> 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, focusing on network interoperability via further adoption of TCP/IP. It was founded by Dan Lynch, an early Internet activist. From the beginning, it was attended by large corporations, such as IBM and DEC. Interoperability conferences have been held every year since then. Every year from 1985 through 1993, the number of attendees tripled.

IBM, ATT and DEC were the first major corporations to adopt TCP/IP, despite having competing internal protocols (SNA, XNS, etc.). In IBM, from 1984, Barry Appelman's group did TCP/IP development. (Appelman later moved to AOL to be the head of all its development efforts.) They navigated the corporate politics to get a stream of TCP/IP products for various IBM systems, including MVS, VM, and OS/2. At the same time, several smaller companies began offering TCP/IP stacks for DOS and MS Windows, such as the company FTP Software, and the Wollongong Group.<sup>[7]</sup> The first VM/CMS TCP/IP stack came from the University of Wisconsin.<sup>[8]</sup>

Back then, most of these TCP/IP stacks were written single-handedly by a few talented programmers. For example, John Romkey of FTP Software was the author of the MIT PC/IP package.<sup>[9]</sup> John Romkey's PC/IP implementation was the first IBM PC TCP/IP stack. Jay Elinsky and Oleg Vishnepolsky of IBM Research wrote TCP/IP stacks for VM/CMS and OS/2, respectively.<sup>[10]</sup>

The spread of TCP/IP was fueled further in June 1989, when AT&T agreed to put into the public domain the TCP/IP code developed for UNIX. Various vendors, including IBM, included this code in their own TCP/IP stacks. Many companies sold TCP/IP stacks for Windows until Microsoft released its own TCP/IP stack in Windows 95. This event was a little late in the evolution of the Internet, but it cemented TCP/IP's dominance over other protocols, which eventually disappeared. These protocols included IBM's SNA, OSI, Microsoft's native NetBIOS, and Xerox' XNS.

## Key architectural principles

An early architectural document, RFC 1122, emphasizes architectural principles over layering.<sup>[11]</sup>

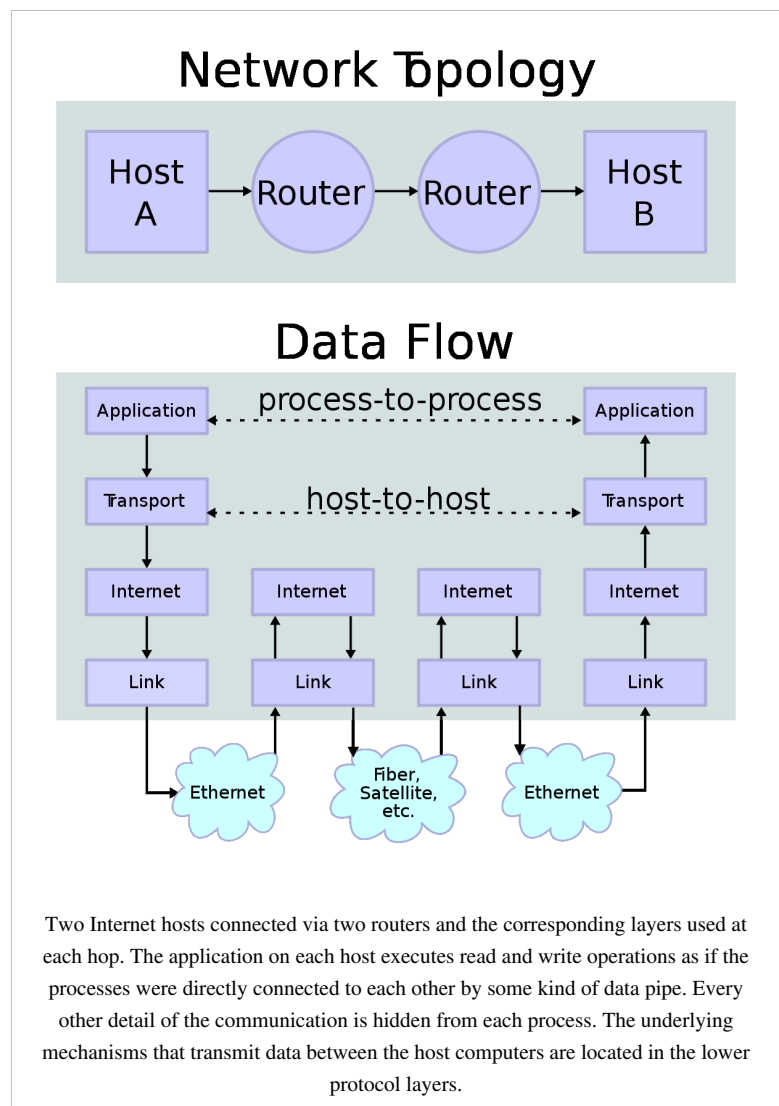
- End-to-end principle: This principle has evolved over time. Its original expression put the maintenance of state and overall intelligence at the edges, and assumed the Internet that connected the edges retained no state and concentrated on speed and simplicity. Real-world needs for firewalls, network address translators, web content caches and the like have forced changes in this principle.<sup>[12]</sup>
- 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)."<sup>[13]</sup> "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."<sup>[14]</sup>

## Layers in the Internet protocol suite

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.

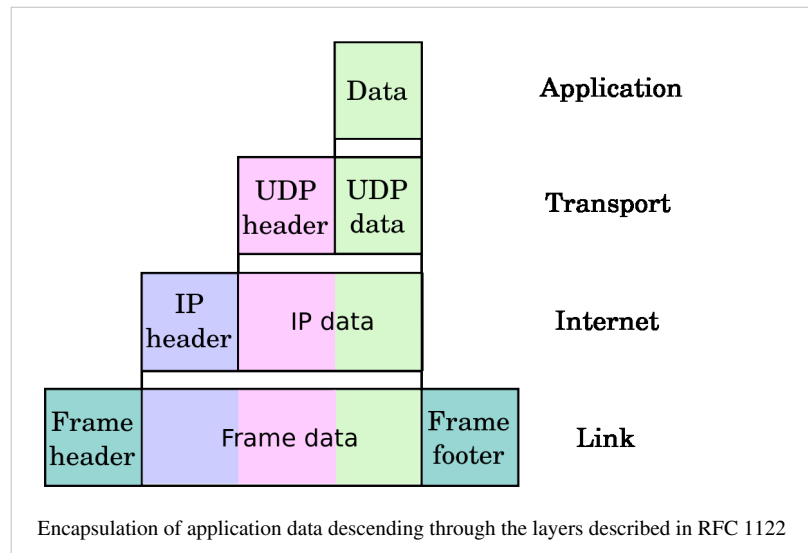
The "layers" of the protocol suite near the top are logically closer to the user application, while those near the bottom are logically closer to the physical transmission of the data. Viewing layers as providing or consuming a service is a method of abstraction to isolate upper layer protocols from 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.

Even when the layers are examined, the assorted architectural documents—there is no single architectural model such as ISO 7498, the Open Systems Interconnection (OSI) model—have fewer and



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.

This abstraction also allows upper layers to provide services that the lower layers cannot, or choose not, to provide. Again, the original OSI model was extended to include connectionless services (OSIRM CL).<sup>[15]</sup> 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).

This model lacks the formalism of the OSI 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 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).<sup>[16]</sup>
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)<sup>[17]</sup>
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 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 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 has the responsibility of sending packets across potentially multiple networks. Internetworking requires sending data from the source network to the destination network. This process is called routing.<sup>[18]</sup>

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 sending packets of data (datagrams) from source to destination by sending them to the next network node (router) closer to the final destination.

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. So, IP can carry data for a variety 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.

The internet layer 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.

## Transport layer

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. Its responsibility includes end-to-end message transfer 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 TCP, or connectionless, implemented in 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 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.

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).

For example, the 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.

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 contains 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).<sup>[19]</sup> Data coded according to application layer protocols are then encapsulated into one or (occasionally) more transport layer protocols (such as TCP or 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).

## Layer names and number of layers in the literature

The following table shows various networking models. The number of layers varies between three and seven.

Kurose, <sup>[20]</sup> Forouzan <sup>[21]</sup>	Comer, <sup>[22]</sup> Kozierok <sup>[23]</sup>	Stallings <sup>[24]</sup>	Tanenbaum <sup>[25]</sup>	RFC 1122, Internet STD 3 (1989)	Cisco Academy <sup>[26]</sup>	Mike Padlipsky's 1982 "Arpanet Reference Model" (RFC 871)	OSI model
<i>Five layers</i>	<i>Four+one layers</i>	<i>Five layers</i>	<i>Five layers</i>	<i>Four layers</i>	<i>Four layers</i>	<i>Three layers</i>	<i>Seven layers</i>
"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"	OSI 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

Some of the networking models are from textbooks, which are secondary sources that may contravene the intent of RFC 1122 and other IETF primary sources.<sup>[27]</sup>

## 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 must impose monolithic architecture above the transport layer. For example, the NFS application protocol runs over the eXternal Data Representation (XDR) presentation protocol, which, in turn, runs over a protocol called Remote Procedure Call (RPC). RPC provides reliable record transmission, so it can run safely over the best-effort UDP transport.

Different authors have interpreted the RFCs differently, about 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.

Several authors have attempted to incorporate the OSI model's layers 1 and 2 into the TCP/IP model, since these are commonly referred to in modern standards (for example, by IEEE and ITU). This often results in a model with five layers, where the link layer or network access layer is split into the OSI model's layers 1 and 2.

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.

The IETF protocol development effort is not concerned with strict layering. Some of its protocols may not fit cleanly into 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".<sup>[27]</sup>

Conflicts are apparent also in the original OSI model, ISO 7498, when not considering 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). GRE uses the same mechanism that OSI uses for tunneling at the network layer.

## Implementations

No specific hardware or software implementation is required by the protocols or the layered model, so there are many. Most computer operating systems in use today, including all consumer-targeted systems, include a TCP/IP implementation.

A minimally acceptable implementation includes the following protocols, listed from most essential to least essential: IP, ARP, ICMP, UDP, TCP and sometimes IGMP. In principle, it is possible to support only one transport protocol, such as UDP, but this is rarely done, because it limits usage of the whole implementation. IPv6, beyond its own version of ARP (NDP), ICMP (ICMPv6) and IGMP (IGMPv6), has some additional required functions, and often is accompanied by an integrated IPsec security layer. Other protocols could be easily added later (possibly being implemented entirely in userspace), such as DNS for resolving domain names to IP addresses, or DHCP for automatically configuring network interfaces.

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. Most IP implementations are accessible to programmers through sockets and APIs.

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.

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 below the link layer, typically using an application-specific integrated circuit (ASIC) chipset for each network interface or other physical standard. High-performance routers are to a large extent based on fast non-programmable digital electronics, carrying out link level switching.

## 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](http://www.columbia.edu/~rh120/other/tcpdigest_paper.txt)). TCP Digest (UUCP). . Retrieved 2007-07-05.
- [7] Wollongong (<http://support.microsoft.com/kb/108007>)
- [8] A Short History of Internet Protocols at CERN (<http://www.weblab.isti.cnr.it/education/ssfs/lezioni/slides/archives/cern.htm>)
- [9] About I "romkey" (<http://www.romkey.com/about/>)
- [10] Barry Appelman
- [11] Architectural Principles of the Internet (<ftp://ftp.rfc-editor.org/in-notes/rfc1958.txt>), RFC 1958, B. Carpenter, June 1996
- [12] 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](http://www.csd.uoc.gr/~hy558/papers/Rethinking_2001.pdf)), Marjory S. Blumenthal, David D. Clark, August 2001
- [13] p.23 INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September 1981 Jon Postel Editor (<http://www.ietf.org/rfc/rfc0791.txt?number=791>)

- [14] Requirements for Internet Hosts -- Communication Layers p.13 October 1989 R. Braden, Editor (<http://tools.ietf.org/html/rfc1122#page-12>)
- [15] [ OSI: Reference Model Addendum 1: Connectionless-mode Transmission,ISO7498/AD1],ISO7498/AD1, May 1986
- [16] "Information processing systems -- Open Systems Interconnection -- Internal organization of the Network Layer" ([http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=16011](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=16011)), ISO 8648:1988.
- [17] "Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 4: Management framework" ([http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=14258](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=14258)), ISO 7498-4:1989.
- [18] IP Packet Structure (<http://www.comsci.us/datacom/ippacket.html>)
- [19] *TCP/IP Illustrated: the protocols* (<http://www.kohala.com/start/tcpipiv1.html>), ISBN 0-201-63346-9, W. Richard Stevens, February 1994
- [20] 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](http://www.pearsonhighered.com/educator/academic/product/0,,0321497708,00+en-USS_01DBC.html))
- [21] 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\\_n83pPt5uCreCih0HnWvNcxg#PPA29,M1](http://books.google.com/books?id=U3Gcf65Pu9IC&printsec=frontcover&dq=forouzan+\))
- [22] 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](http://books.google.com/books?id=jonyuTASbWAC&pg=PA155&hl=sv&source=gbs_toc_r&cad=0_0&sig=ACfU3U18gHAia1pU_Pxn-rhkCnH1v70M6Q#PPA161,M1))
- [23] 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=ACfU3U3ofMwYAbZiGz1BmAXc2oNNFC2b8A#PPA129,M1](http://books.google.com/books?id=Pm4RgYV2w4YC&pg=PA131&dq=TCP/IP+model+layers\))
- [24] 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=ACfU3U38aXznzeAnQdbLcPFXfCgxAd4IFg](http://books.google.com/books?id=c_AWmhkovR0C&pg=PA35&dq=internet+layer+\))
- [25] Andrew S. Tanenbaum, Computer Networks, Prentice Hall 2002, ISBN 0-13-066102-3 ([http://books.google.com/books?id=Pd-z64SJRBA&pg=PA42&vq=internet+layer&dq=networks&hl=sv&source=gbs\\_search\\_s&sig=ACfU3U3DHANelz0sOsd5NK4VXSrgNFYVAw#PPA42,M1](http://books.google.com/books?id=Pd-z64SJRBA&pg=PA42&vq=internet+layer&dq=networks&hl=sv&source=gbs_search_s&sig=ACfU3U3DHANelz0sOsd5NK4VXSrgNFYVAw#PPA42,M1))
- [26] Mark Dye, Mark A. Dye, Wendell, Network Fundamentals: CCNA Exploration Companion Guide, 2007, ISBN 1-58713-208-7
- [27] R. Bush; D. Meyer (December 2002), *Some Internet Architectural Guidelines and Philosophy* (<http://www.ietf.org/rfc/rfc3439.txt>), Internet Engineering Task Force,

## 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](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/>)
  - 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](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
-

# Article Sources and Contributors

**IEEE 802** *Source:* http://en.wikipedia.org/w/index.php?oldid=508517875 *Contributors:* 2001:630:12:107B:D685:64FF:FE0F:919A, 2ndjpeg, Aarktica, Aka, Alinja, Asbestos, Bpmullins, C-M, Calicot, CanadianLinuxUser, Casey Abell, Cmdrjameson, Conversion script, DaniRoisman, Debresser, DenesVadasz, Diagraph01, Discospinster, Djg2006, Don40f4, Etienne.navarro, Flashpoint145, Freedomlinux, FunPika, Gowr, Graham87, Grubber, Guy Harris, Hede2000, Hellisp, Hydrogen Iodide, Int21h, Integr, Itai, J G Campbell, JamesBWatson, Jiraffe, JonHarder, Jtk, Kirils, Kozuch, Kvng, LA2, Lcabanel, Lumag Lumag, Materialsscientist, Matt Britt, Maxwell Rain, Nasa-verve, Nathan8225, Nil0lab, Nixdorf, Nsaa, Nubiotech, OS2Warp, Oli Filth, Plodder81, Qinghao, Rapsar, Raveeshworldwide, Rick Sidwell, Rigger151, Rmurias, Roux-HG, Rwww, Scruff323, Sligoeki, Sonia, Steven Hepting, Terra Xin, The Anome, Timwi, Tsystems, Tyler, Ugur Basak, Versus22, Vishnava, Wassini, Wrs1864, X746e, 127 anonymous edits

**Ethernet** *Source:* http://en.wikipedia.org/w/index.php?oldid=516984316 *Contributors:* 0612, 121a0012, 1llovegal, 1stJahman, 4a6f656c, 7, ARC Gritt, Aapo Laitinen, Abdull, Adashiel, AdjustShift, Adrian.benko, Aeluwas, Ahoerstemeier, Ajraddatz, Akriasas, Alan Liefiting, Alansohn, Aldie, Ale jrb, Alecv, Algocu, AlistairMcMillan, Altermike, Alvestrand, Alyssapvr, Amatulic, Ameliorate!, Amillar, Amore proprio, Anaraug, Andareed, Andreas Toth, Andy Dingley, Andybryant, Andyhodapp, Aneah, Angela, Angusmca, Anss123, Applemacintosh10, Arch dude, Arkrishna, Armando, ArnoldReinhold, Artaxiad, AtomEdge, Avono, Azaghal of Belegost, B4hand, BD2412, Bbachrac, Bctwriter, Bdmcmahon, Becritical, Betacommand, Bestbest1, BiT, Bidabadi, Bigboymcgee, Bigjim, Bilbo1507, Biot, BlueAg09, Blutrot, Bobblewik, Boffy b, Bomac, Bookbrad, Bookofjude, Bootleggingly mcbootleg, Boscobiscotti, Bovineone, Branclem, Branko, Brian Patrie, Bruce lee, Btlm, Bumml3, Buy P%E%P%\$%I, C0nanPayne, CRGreathouse, CS46, Calltech, Caltas, CamTarn, Can't sleep, clown will eat me, Capricorn42, Carlo.lerna, Casey Abell, Catgut, Causantin, Cburnett, CesarB, Chaosgate, Charles Gaudette, Chasingsof, Chealer, Chmod007, Chowbok, Cjdaniel, ClementSeveillac, Closedmouth, CloudNine, Cluth, Cmdrjameson, Colin Marquardt, Conversion script, Corpx, Corti, Corwin8, Cpl Syx, Creidieki, Crispmuncher, Crissow, CryptoDerk, Cstanners, Cybercobra, DvDm, Daa89563, Dachshund, Damian Yerrick, Davehard, David Biddulph, DavidBailey, DavidH, Dcorzine, Deflective, Dennis Brown, Dicklyon, Digitalsushi, Discospinster, Diskdoc, DocWatson42, Dogcow, Donniecelofarrow, Download, Drak2, Drphilharmonic, Dtcddthingy, EEgiri18, EagleOne, Ebear422, Ecnrad, Eekster, Efa, Egil, Ehn, Eivind F Øyangen, El Cubano, ElBenevolente, Electron9, Elektrik Shoos, Eleuther, Elinruby, Eloil, Engineerism, Enjo4586, EoGuy, Epr123, Evil genius, Ewlyahoooom, Excirial, Facts707, Femto, Fenrisulfur, Fetofs, Fieldday-sunday, Finest1, Firsfron, Flewis, FocalPoint, Frankchn, Frap, Fratrep, Fred Bradstadt, Fredrik, Fresheneesz, Fudoreaper, Fulldecnt, Furrykef, G4wsz, GTAKillerEric, Gah4, Gascreed, Geek2003, Gekkoabaster, Generica, Ghaly, Giftlite, Glacialfox, 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, Iamzemasteraff, Ibc111, Idkydidthis, Igoldste, Indefatigable, Integr, Irdescent, Irq, Isaac Rabinovitch, Itai, Itusg15q4user, J.delanoy, J04n, Jakohn, JamesBWatson, JamesEG, JeLuF, Jec, Jemichel, Jhartmann, Jim.henderson, John a s, JohnOwens, JohnWittle, Johnblade, Johnnyboyshoots, Johnnue, JonHealer, Chmod507357, Joyous!, Jtkiefer, Juansempere, Judzillah, Juliancolton, K45671, KGasso, Kakomu, KaragouniS, Karn, Kasabax, Kate.woodcroft, Khb3rd, Kbolino, Kbrose, Keith D, KelleyCook, Kerotan, Kevmcs, Kgfeischmann, Khalad, Kinema, King of Hearts, Kjkolb, Knuemo2, Ksn, Kuru, Kvng, Kwamikagami, Kwiki, Lallolu, Lamro, Laudaka, Lavenderbunny, Lee Carre, Leebo, Leotohill, Lightmouse, Ligulem, LilHelpa, Limbo socrates, LittleBenW, LittleOldMe, Lockg, Lolpack, 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, Mightymys, Mike Rosoft, Mikm, Milan Keršláger, Mindmatrix, Minesweeper, Mmiszka, Modulatum, Mortense, MoussePad, Moxfyre, Mozzerati, MrFish, MrOllie, Mrand, Mrh30, Mwanner, Myanw, N TRoPY, NJM, NawlinWiki, NeaNita, Nelson50, NewEnglandYankee, Nicolaasuni, Nightraider0, Nikevich, Nimiew, Nishkid64, Nixdorf, Nnemo, Norm, Northamerica1000, Now3d, Nroets, Nthep, Nubiotech, Nvj, Nyttend, Oakad, Ohconfucius, Ohnoitsjamie, Oneocean, OverlordQ, OwenX, Oxymoron83, Packetslinger, Palmer1973, Paul, Paul Koning, Peck123, Pekaje, Percy Snoodle, Peruvianlama, Petr, Peyre, Pgalert, 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, Rholtton, Rich Farmbrough, RichardBennett, Rick Sidwell, Ricsi, Rjwilmsi, Rlcantwell, RoySmith, Roybadami, Royote, Rror, Rsrikanth05, Rufus210, Ruud Koot, Rwn4nd4, Rwww, SF007, SHCarter, Salsa Shark, Sam Hovecar, Sander123, Sceptre, Schneelocke, Scott McNay, ScottJ, SeaChanger, Selmo, Setu, Sfisher, Shadowjams, Shanes, Sharkford, Shieldforyoureyes, Sietse Snel, Silenceisof, Simoscie, Sligoeki, SoSaysChappy, Sobelbob, SpamBilly, Spinitia, Steev, Stephan Leeds, Stephenb, Steven Luo, Stratocracy, Stuart P. Bentley, Suruena, Svick, Swadpick, Swarnabhra, TAnthony, Tas50, Tedernt, 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, Tmontalv, TomPhil, Tonsofpcs, Tooki, Tothwolf, Tpbardbury, Transmission 1000, Trev M, Tunnie, Turgan, Tverbeek, Typo47, Tyz, UU, Ukexpat, FormulaX, UncleBubba, Unschool, Urvabara, Useoth, Utiuew, Vashiti, Vdm, Veinor, Versus22, Vidmes, Vijaykumar, Viridae, Vmenkov, W Nowicki, WaZise, WadeSimMiser, Wasisnt, Wattlesmorse, Wavelength, Wayfarer, WaysToEscape, Wbenton, Wbm1058, Wefa, WhiteDragon, Wik, WikiJaZon, Wikiborg, Wikifranz, WillAndrews, WillLord, Willy on Wheels over Ethernet, Wimt, Wingnutjam, Wk muriithi, Wmaham, Wmasterj, Woohookitty, WriterHound, Wrs1864, Wtshymanski, Xenium, Xnatedawgx, Yintan, Yudiweb, Yuvalnod, Yyy, ZenerV, Zoro10ne, Zhangyue, Zodon, Zoicon5, 1009 anonymous edits

**Ethernet frame** *Source:* http://en.wikipedia.org/w/index.php?oldid=517526905 *Contributors:* Alexh19740110, Bostonvaulter, Dewritech, Electron9, EoGuy, Glenn, Guy Harris, Gyro Copter, Hawaiian717, I340793, Integr, Jiemingwei, Kvng, M1ss1ontomars2k4, Magioladitis, Mange01, Mayur, Ocdex, QueBurro, Quinn d, Shashikills, Staaki, Systemparadox, Tchai, Tomaxer, Torzsmokus, W Nowicki, Wjwatson67, Xezbeth, Yemisilin, 83 anonymous edits

**IPv4** *Source:* http://en.wikipedia.org/w/index.php?oldid=517827878 *Contributors:* -Majestic-, A.R., Abdull, Acather96, Adrian.benko, AlephGamma, Alexkon, AlistairMcMillan, Althena, AndreasWittenstein, Andrewmc123, Andypar, Angela, ArglebargleIV, Arjayay, Armando, Axelrivar, Barro, Barryd815, Begoon, Bmpercy, Borgx, Breno, Brest, Bro1960, Brouhaha, C. A. Russell, CCFreak2K, CWii, Calcwatch, Calinou1, CaribDigita, Carlo.arenas, Cburnett, CecilWard, CesarB, Chris D Heath, Chrishmt0423, Christan80, Cm115, Cmicahel, Conversion script, Coredesat, Corti, Crashdooom, Cwolfoosheep, Cybjit, Cynical, DARTH SIDIOUS 2, DBigXray, DH85868993, Dan6hell66, Daniel Staal, Daniel.Cardenas, Danielbarnabas, DataWraith, DavidDelaune, DenisKrivoshchev, Dmaftei, Dnas, Dotshuai, DreamGuy, Dsearls, Duffman, Dunganb, Dungal, Ed g2s, EdC, Ekspiulo, El C, Electron9, Enjo4586, Ericbarch, ErikWarmelink, EvilSS, Exallium, Faco, Favoniasn, Floydpink, FormulaX, Fred Bradstadt, Fredrik, Fresheneesz, Gallando, Garo, Gehlers, General Wesc, Giftlite, Glrx, Graciella, Graham87, Gravavchaudhary, Gthm159, Hairy Dude, Hcbkowitz, ILike2BeAnonymous, IRedRat, Ilario, Imroy, Indrek, Ironholds, JTN, Jasper Deng, Jbergste, Jbohac, JeroenMassar, Jesant13, Jgeer, Jk2q3jrklse, Jnc, Joelby, Johnnyboyshoots, Johnstlade, Joseph Solis in Australia, Joshua, Jpyeron, Jwdonal, Kaal, Kaeso, Karada, Kasperl, Katieh5584, Kbrose, Kevin66, Kgfeischmann, Khr0n0s, Khvalamde, Kickboy, Kiore, Kmwiki, Krellis, Kvng, Kwamikagami, Kyng, Leuqarte, Liface, Lightmouse, Lph, Lukerichita, M.O.X, MECU, Magioladitis, Magnus.de, Maimai009, Mange01, Marcoscm, Markrod, Melnakeeb, Mewashere1, MiNeEstasVortaristo, MichaelGoldshteyn, Milan Keršláger, Mindmatrix, Miss Saff, Mmtrebuchet, Mochi, Mokgen, Molerat, Morten, Mushroom, Nathan Hamblen, Nealmb, Neelix, NewEnglandYankee, Nil Einne, Noldoaran, Nubiotech, Ojw, Omniplex, Onceler, Opelio, Outback the koala, Parent5446, Parkamark, Paul, PaulHanson, Pengo, Peterhoneyman, Phantomdj, Philip Trueman, Phoenix314, Phorque, Piano non troppo, Pmj, Pratikarun, Presto8, Ptmc2112, Raanoon, Ranto, Rantsroamer, Rbhjhm, Rchandra, RevRagnarok, RexNL, Rjwilmsi, Robert Brookway, RoySmith, Rpwodbu, Ruwolf, RxS, RyanWKeen, Sarafankit, Scjessey, Seaphoto, Shane kerr, Simon J Kissane, Sirmelle, SmilingBoy, Smurrayinchester, SpacemanSpiff, Spearhead, SpectatorThid, SpeedyGonsales, Stephan Leeds, StrangerInParadise, Suruena, Swellesley, Taestell, Teemunk, Tenrenieht, The Anome, The Thing That Should Not Be, TheGreyArea, Themonstergirla, Rhadu, Tide rolls, Toolnut, Tyler.szabo, UU, Ultimud, Undereference, Versus22, Visiting1, Vivio Testarossa, Vkartik, Wavelength, Winston Chuen-Shih Yang, Wolfsbane2k, Woohookitty, Wrs1864, Wyksztalcioch, XavierHager, Xibe, Yann Lejeune, Yyy, Zanetu, Zetawoof, Zfr, ^demon, 518, עֶשֶׂה נָתַן anonymous edits

**User Datagram Protocol** *Source:* http://en.wikipedia.org/w/index.php?oldid=514987674 *Contributors:* 1exec1, 28421u2232nfencenc, A5b, Aapo Laitinen, Alfio, Ali@gwc.org.uk, Alison22, Alvin-cs, Andrew Kember, Aram33, Ardonik, Areh adeola, Ashmo, Ayeroxor, BAxelrod, Beacon11, Beefman, BenAveling, Bender235, Benqmonitor, Bentogoa, Bobby Caudwell, Bobstopper, Borgx, Breno, Bryan Derksen, Burisch, Cassan, Cburnett, Chenzw, Chiefmanzzz, Cincapatinr, Conrad.Irwin, Conversion script, CraSH, CrookedAsterisk, Cvalda, DSat, Damian Yerrick, Daniel Staal, DavidSol, DeadEyeArrow, DennyColt, DesTroy, Dfurbeck, Dicentra, Djordjes, Dkogh, E Wing, E090, Eagleal, Eclipsed, Enjo4586, Erik Garrison, Esmond.pitt, Falcon9x5, Fennec, Fightin' Phillie, Flewis, Frankie, FrisoB, Frodet, Fromagesteiel, Fubar Obfusco, Fudgefr, Gamera2, Giftlite, Ginkgo100, Gniphallir, Golwengaud, Gordoni, Gorryf, Gracefool, Graham87, Gsmgm, Guy Harris, Hdante, Headbangerkeny, Hitherebrian, I dream of horses, IRedRat, Icestorm815, Ideoplex, InverseHypercube, Irrypride, Itai, JNighthawk, JTN, Jengelth, Jjinjo, Jncraton, Joachim Wuttke, Joanjoc, Johnny O, Johnuniq, Jtk, Jóna Þórunn, Kbrose, Kgfeischmann, Kinema, Koviany, Krellis, Kvng, Kwamikagami, Kyng, Lissajous, LI1324, Locos epraix, MLeb, Mahjongg, Mam711, Mange01, Mani1, Materialsscientist, Matt Darby, Michael Zimmermann, Mike Dill, Mike6271, Misterdom, Modulatum, Mpeg4code, Mwholt, Neile, Nixdorf, Nroets, Nubiotech, Od Mishehu, Oumitch, Oxymoron83, PatheticCopyEditor, Paulish, PedroPVZ, Pelesl, Petri Krohn, Phatom87, Plindemann, Plugwash, Pocuscualin, Public Menace, Quiddity, R2richar, Razorflame, Recurring dreams, RedWolf, Rememberway, Reub2000, RevRagnarok, Reywas92, Rick Sidwell, Robbe, RobertL30, Romanm, Ronz, Rotlink, Rwww, S-n-ushakov, SWAdair, Samersarhan83, Sammydee, Samuel, Sci13960, Smeggysgem, Spearsall, S2393, Suruena, Tdangkhao, Teapeat, TechyOne, Teemuk, Teh roflmaor, Template namespace initialisation script, Terrycoons, The Anome, The Monster, The Utahraptor, The-tenth-zdog, Theopolism, Thray, Tjdw, Tobias Bergemann, Towel401, Tpbardbury, Twinxor, UncleBubba, Updayayakc, Vanished user 5zariu3jisj0j4irj, Vinu Padmanabhan, Voidxor, WestwoodMatt, Wiarthurhu, Wingman417, Wkcheang, Xezbeth, Yeloyakit, Zara1709, ^demon, Александр, Тиверополник, 352 anonymous edits

**Transmission Control Protocol** *Source:* http://en.wikipedia.org/w/index.php?oldid=516667414 *Contributors:* 10metreh, 2620:0:102C:9:221:6AFF:FE6F:2758, 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, Anna Lincoln, AnotherNitPicker, Anwar saadat, Anwar saadat, Aravachaudhary, Arnhemcr, Arsenal9boi, Asenine, Ashwinsbhat, Astronomerren, Avono, Awy997, B4hand, BAxelrod, Banej, Beccus, Beland, Betterworld, Bezenek, Bigdumbdinosaur, Bilbo1507, Bill Malloy, BillMcGonigle, Biot, Bkell, Bkkrad, Blacksqr, Blanchardb, Boothy443, Boscobiscotti, Brech, Breno, Brighterorange, Brion VIBBER, Bstrand, Bubba hotep, Butros, C10191, Can't sleep, clown will eat me, CanadianLinuxUser, Camellus, Cbretin, Cburnett, Centrex, Charles.partitionlow, Chealer, Chelurashka, Christopher P, Cincapatinr, 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, Dori, DrHannibal216, Drake

Redcrest, Duckbill, DylanW, Egg, Ego White Tray, Eirik (usurped), El pak, EncMstr, Encognito, Enjoia4586, Enviroboy, Epr123, Equendil, Eric-Western, Erik Sandberg, Esmond.pitt, Evices, Evil Monkey, Explicit, FDD, FGont, Fabiob, Fawcett5, Fcp999, Fishal, Fisherisland14, Fleminra, Foobaz, Fred Bradstadt, Frederico1234, Fredrik, Fredrikh, Frenchigh, Fresheneesz, Filer, Fubar Obfusco, Gaius Cornelius, GalaxiaJuy, Gamera2, Garion96, GarryAnderson, Ghetoblastian, Giftlite, GilHamilton, Glenn Willen, Gmaran23, Gmaxwell, GoingBatty, Goplat, Graeme Bartlett, Graham.Fountain, Graham87, Guy Harris, Gwinnadain, Haggis, Hairy Dude, Hamtechperson, Harput, HarrisonLi, Harryzilber, Harvester, Helix84, Henrikudborg, HiB2Borno2B, Hilgerdenaar, I already forgot, I-baLL, IOLJeff, IRP, Idcmp, Ideoplex, Iitywybmad, Imcdnzl, Imroy, Inter, Ingrid, Inwind, Izno, J. Nguyen, JCO312, JTN, Jaan513, Jackol, Jcarlos-causa, Jec, JeffClarkis, Jehorn, Jesant13, Jergobac, Jfmanets, Jgeer, Jgrahn, Jinno, Jnc, JoanneB, Jagers, John Vandenberg, JohnTien, JohnHarder, Jonathan Hall, Jondel, Jowagner, Jsavage, Jtk, Juliancolton, JuneGloom07, Jusdafax, Jxw13, Karada, Kartano, Karthick.s5, Kasperl, Katimawan2005, Kbrose, Kenyon, Kevinmon, Kgfleischmann, Kim Bruning, Kinema, Kluhullier, Krellis, Kubanczyk, Kumarat9pm, Kvgng, Kwamikagami, Kwiki, L Kensington, LachlanA, Lam Kin Keung, Lanilsson, Lark ascending, LeiZhu, LeoNomis, Leolaursen, Leszek Jafczuk, Leyo, LiDaobing, Lightdarkness, Lights, LilHelpa, Lilac Soul, Lime, Logicwax, Lokacit443, Lone boatman, 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, Mortense, Mozzerati, Mr Echo, MrBoo, MrOllie, Mrzehak, MtB, Mtsz, N328KF, NKarstens, Nateshwar kamlesh, Nave.notnilc, NawlinWiki, Nayuki, Nealcardwell, Nedim.sh, NeilGoneWiki, Neo139, Neshom, Nestea Zen, Ngriffeth, Nifky7, Nikhil raskar, Nikola Smolenski, Nimiew, Nixdorf, Nk, Nneuman, Northamerica1000, Nubiatech, Nviladkar, Ocaasi, Ojs, Okiefromokla (old), Oldiowl, Oli Filth, Omicronperseid8, Ordoon, Ortzinator, OsirisV, Otets, Ouimetch, Oxyacanthon, Oxyamoron83, P0per, PBSurf, PS3ninja, Padmini Gaur, Pak21, Palica, PanagosTheOther, Papadopa, Pde, PedroPVZ, Pegasus1138, Perfgeek, PeterB, Peytons, Pfalstad, Pg8p, Pgr94, Phandel, Phantomsteve, Phatom87, PhilKnight, Phuyal, Jtk, Juliancolton, JuneGloom07, Plugwash, Pluknet, Pmadrid, Poelq, Prakash mit, Prasannaxd, Prashant.khodade, Prunk, PureRumble, Purplefeltangel, Putdust, PxT, Quantumobserver, Quibik, Qutezuze, RA0808, Raul654, Rdome, RedWolf, Rettetast, RevRagnarok, Revolut, 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, Sethwm, Sh manu, Shaddack, Shadownjan, Shadowjams, Shultzc, Sietse Snel, Silverwindx, Sim, Sleigh, Smappy, Smsarmad, Smyth, Snowwolf, Some Wiki Editor, Spearhead, SpeedyGonsales, Splint9, Spoon!, Squidish, StephenHemminger, Stephenb, Stevenwagner, Stonesand, StradivariusTV, StubbyT, Suruena, Svinodh, SwisterTwister, Synchrite, Syp, THEN WHO WAS PHONE?, Talel Atlas, TangentCube, Tariqajbotu, Tasc, Technobadger, TeelousFellow, Teles, Temple namespace initialisation script, Tfl, The Anome, The Monster, The Thing That Should Not Be, The-tenth-zdog, TheVoid, Thine Antique Pen, Tide rolls, Timotheus Canens, Timwi, Toh, Tomchiuke, Tommy2010, Torla42, Trou, TuukkaH, Umar420e, UncleBubba, Unixguy, Urmajest, Ursushoribilibi, Uruiamme, Vedantm, Velella, Via strass, Vinu Padmanabhan, Vrenator, WLU, WikHead, WitiLaurent, Wimt, Wkcheang, Wlgrin, WojPob, Wolfkeeper, Woohookitty, Wrs1864, Xaphnir, Ymiaji, Yonatan, Youpilot, Ysangkok, Yyy, Zachlipton, ZeroOne, Zeroboo, Zundark, Zvar, شات انوميز, 1136 anonymous edits

**OSI model** *Source:* <http://en.wikipedia.org/w/index.php?oldid=517358512> *Contributors:* 0612, 0x6D667061, 1337 JN, 1966batfan, 24.12.199.xxx, 28bytes, 336, 63.227.96.xxx, 7, 75th Trombone, 802geek, 90 Auto, @modi, A412, A930913, ABF, Abarry, Abune, Adamantios, Addshore, Adibob, Adityagaur 7, Adj08, Adoniscik, Adrianwn, Advancedtelcotv, Ageekgal, Ahoersteimeier, Aitias, Ajo Mama, Ajw901, Alansohn, Albanaco, Aldie, Ale jrb, AlistairMcMillan, Allens, Alphachimp, Alucard 16, Alvestrand, Amillar, Amithabia76, Amantoli, Andjohn2000, Andre Engels, Andybryant, Angryscorpio, Animum, Anjola, Ankhhorpork, Anna Lincoln, Anon lynx, Anonymous anonymous, Another-anomaly, Apocryphite, Apparition11, Arroww, Artur Perwenis, Arunachalamanoahar, Ashutosh.mcse, Aslambasha09, Asn1tlv, AtomicDragon, Atreyu42, Audunv, Avitesh, AxelBoldt, Ayengar, B4hand, BACbKA, BDerry, Bakilas, Balajia82, Bariswheel, Bchiap, Bdamokos, Beelaj, BenLiyanage, Beno1000, Biblbrosks, Bjelleklang, Blech, Buickies238, Bmylez, Bobo192, Bogdangiusca, Boikej, Bojer, BommedDing, Bonobosaner, Booyabazooka, Borgx, Brambleclawx, Brandon, Brick Thrower, Brougham96, Bryan Derksen, BuickCenturyDriver, Bzimage.it, Bücherwürmlin, CDima, Cioeland, 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, Chirag, Chrislk02, Chupon, Cicikdrager, Citicat, Closedmouth, Cokoli, Cometstyles, Conquest ace, Conversion script, Corion, Courcelles, Cputdrc, CraSH, CraigBox, Crasheal, Crimsonmargarine, Cs mat3, Ctibolt, Cxxl, Cybercobra, CyborgTosser, Cyktsui, CynicalMe, DARTH SIDIOUS 2, DJPohly, DSParillo, Damian Yerrick, Daniel, Danlev, Dave2, Davetrainer, David Edgar, David Gerard, David0811, DavidBarak, DavidLevinson, Davidjd, Dcooper, Dcovelt, Deagle AP, Delfeye, Deildot, DeltaQuad, Demitsu, Denisarona, 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, Edivorcer, Edward, ElKevbo, Eldiablo, Eleassar, Elfosardo, Eliezerb, Elipongo, Emperorbma, EnOreg, Enjoia4586, Enochlaur, Epr123, Eric Soyke, Everyking, Evillawngnome, Ewlyahocoom, Excirial, FF2010, Falk.H.G., Fang Aili, Feezo, Fiable.biz, Filemon, Finlay McWalter, Fjpanna, Fleg31789, Flewis, Flowanda, Fraggel81, FrankTobia, Fred Bradstadt, Fredrik, Free Bar, FreshPrinz, Fresheneesz, Friday, Friedo, Frigintator, Fullstop, Fumitoli, Fuzheado, Fw, Fw, GDonato, GGShinobi, Gadium, Gafex, GarestGilson, 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, Hcbekrowitz, Hdante, Helix84, Hellomarias, Henrikholm, Herbee, Heron, Hes Nikke, Heter, HexaChord, Hgerstung, Hiddekel, Highbpriority, Honeymey, I dream of horses, IMSoP, IReceivedDeathThreats, Iambk, Iambossaghari, Ideoplex, Ifroggie, Ilario, Immunize, Inkhorn, Inkling, Insideratelymnn, Intgr, Inversetime, InvisibleK, Inwind, Iridescent, Imavash, IronGargoyle, Ishikawa Minoru, Island Monkey, Isofox, Isthisthingworking, Itpastorn, Itusg15q4user, Iviney, J.delanoy, Jmatthews, JV Smithy, Jake Wartenberg, JamesWatson, Jannetta, Jatinchina, Jauerback, Jchstrn, Jcw69, Jdrmk, JfTataMe, Jeanjour, Jeff G., Jeffrey Mall, Jessemerriman, Jetatusk, Jhilverj, JidGom, Jim1138, Jimw338, Jgiclns5123, Jmorgans, Jnc, JoanneB, JodyB, Joebeone, John Hopley, John Vandenberg, John254, Johnblade, Johnleemk, Johnuniq, JohnHarder, Jonathanwagner, Jonwatson, Joodas, Josef Sábl cz, Josh Parris, Jovianeye, Joy, Jpta, Jrodor, Jschnur, Jschoon4, Jsonheld, Jusdafax, Kaaveh Ahangar, Kallaspriit, Karekikc, Karpouzi, Kaszeta, Katalaveno, Kaz219, Kazrak, Kbrose, Kcordina, Kerry Veenstra, Kesla, Kevin Rector, Kgrg, Khat17, Killiondude, Kim Rubin, Kingpin13, Kirill Lokshin, Kkbairi, KnowledgeOfSelf, Kraftlos, Kramerino, Krampo, Krellis, Kuru, Kvgng, Kyllis, LOLL, LOTRrules, Lachlancooper, Lankiveil, Lawrence Cohen, Lazarus666, Leafyplant, Lear's Fool, Lectornar, Lee Carre, Lighthouse, Lights, LittleOldMe, LittleWink, LizardJr8, Lockcole, Logictheo, Logthis, Lomn, Looxix, Lord Chamberlain, the Renowned, Lordeasav, Lotje, Lulu of the Lotus-Eaters, Luna Santin, Lupin, Lynallendaly, M, MBisanz, MER-C, MIT Trekkie, Maguscrowley, Mahanga, Mahesh Sarmalkar, Majorly, Mange01, Manishar us, Marek69, MarkSutton, MarkWahl, Markb, Markhurd, Markolinsky, MartinHarper, Martinkop, Marvin01, Materialscentist, Mattalyst, Matthew Yeager, Mattjalloway, Mattmill30, Mbc362, Mboverload, McGinnis, Mcnuttj, Mdd, Meepster, MelbourneStar, Mendel, Mephistophelian, Merlion444, Metaclassing, Micahcowan, Michael Hardy, Michael miceli, Mike Rosoff, Mikel Ward, Mikeo, Mikeyh56, Milind m2255, Minimac, Mkweise, Mlewis000, Mmceerman, Mmernex, Mmmeg, Mobius R, Mohitjoshi999, Mohitsport, Mojalefa247, Monterey Bay, Morten, Moxfyre, Mr Elmo, Mr Stephen, Mr.gliban, MrOllie, Mrankur, MrsValdry, Mtd2006, MuZemike, Mudasir011, Mulad, Mwtoews, Myanw, Myheadspinscircles, N-Man, N5iln, Naishadh, Nanshu, Naohiro19, Naresht jangra, Nasa-verve, Natarajuab, Nate Silva, Nathanashleywild, NawlinWiki, Nbarth, Nbhatala, Neevan99, Nejko, Nemesis of Reason, Nethgib, Netsnipe, Niaz, Nick, Nickshanks, Nicolas1981, Nisavid, Nitecruzr, Nivix, Nk, Nkansahreford, Noahspurrier, Nolyann, Northamerica1000, Nsaa, Nubiatech, NuclearWarfare, Nux, OSUKid7, Octahedron80, Odie5533, Ogrest, Oita2001, OIEnglish, Omicronperseid8, Orange Suede Sofa, Ore4444, Originalharry, Ott, Ottosmo, Ouishoebane, Oxyamoron83, PGWarg, Palltrast, Pamri, Panser Born, Paparodo, Parakalo, Pastore Italy, Patch1103, Patrikar, Patstuart, Paul August, PaulWIKIJeffrey, Payal1234, Pb30, Peni, Penno, Pethr, Petrb, Phatom87, Phil Boswell, Philip Trueman, PhilipMW, Pluyo8989, Pmkorkert, Pointillist, Postldf, Postmortemjapan, Praggu, ProPuke, Pseudomonas, Psiphior, Public Menace, Puchiko, Puckly, PyreneisJIM, Pytom, RainbowOfLight, Raju5134, RandomAct, Ravikiran r, RazorICE, Rcannon100, Rebroad, Recognizance, RedWolf, Reedy, Rejax, Rettetast, Rfc1394, Rfclchrist, Rhobite, Rich Farmbrough, RichardVeyard, Richawles, Rick Sidwell, Rjgodo, Rjstynic, Raager, 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, Saintfinds, Sakurambo, Savh, SaxicolousOne, Scarian, Schumi555, Scientus, Scohous, Scolob, Scottonssocks, Seaphoto, Sesu Prime, Shadow1, Shadowjams, SharePointStacy, Shell Kinney, Shirik, Shoefeldesh, ShornAchesons, Shradha deshmukh, Shrofami, Sietse Snel, Simofinl, Simple Bob, SineChristoNon, Sir Nicholas de Mimsy-Pordown, Skier Dude, Sliceofmiami, Slobertson, Smalljim, Smokizy, SnowFire, Snowwolf, Soosed, Sp33dyphil, SpaceFlight89, Speaker to Lampposts, SpeedyGonsales, Spitfire8520, SpuriousQ, Sridev, StaticGull, Stemonitis, Stephan Leeds, Stephen Gilbert, StephenFalken, Stageave, Steven Zhang, StuartBrady, Subfrowns, Sunilmalik1107, Suruena, Suyashparth, Swapouch, Syntaxsystem, TAS, THEN WHO WAS PHONE?, Tagishsimon, Tangotango, Tarekadi, Taruntan, Tbsdy lives, Tcnvc, Techtoucian, Tedickey, Tellyaddict, Tempodivalse, The Anome, The Athlon Duster, The Haunted Angel, The Thing That Should Not Be, Theopolisme, Therumakna, Thief12, Thingg, Think4mit, ThreeDee912, ThunderBird, Tide rolls, Tim Q. Wells, TinyTimZamboni, Tom harrison, TomPhil, Tommy2010, Tompsci, Tony1, Tooki, Tpbabury, Tpvibes, Tranzz, Travelbird, Tree Biting Conspiracy, Trevor MacInnis, Triona, TripleF, Triwbe, Troy 07, Turb0chrq, Tyler.szabo, UU, Umair ahmed123, Uncle Dick, Unknownid123, Vegaswikian, Venu62, Versus22, VidGa, Vishnava, Visor, Vk anantha, Vmгурuprasath, Voidxor, WLU, Waggars, Warrierrakesh, Wyfarer, Weregibil, Whitejay251, WikiDan61, Wikipelli, William Avery, Willking1979, Wilson.canadian, Wily duck, Wingman417, Winston Chuen-Shih Yang, Wire323, Wireless friend, Wishingtown, Wizardist, Wknight94, WoKiCK, Woohookitty, Wrlce, Wrs1864, Wtmitchell, Wshymanski, Yamamoto Ichiro, YamiKaitou, Yamiike, Yms, YolanCh, Yuokool12, ZX81, ZachPruckowski, Zachary, Zoobee79, ܙܘܒܝܐ, Țuman, 3282 anonymous edits

**Internet protocol suite** *Source:* <http://en.wikipedia.org/w/index.php?oldid=515653549> *Contributors:* 130.243.79.xxx, 203.109.250.xxx, 213.253.39.xxx, 66.169.238.xxx, ASUDI, Aapo Laitinen, Abdullah, Abdullais4u, Acceptus, Acroterion, Aeonx, Ahoersteimeier, Alansohn, Albanaco, Aldie, Ale2006, Alek Baka, AliMaghrebi, Aliaspr, Alireza.usa, AlistairMcMillan, Amungale, Ana Couto, Aneah, Anna Lincoln, Anon lynx, Anorom, Arcenciel, ArchonMagnus, Arteitle, ArticCynda, Avant Guard, Avicennasis, Axxess, AxelBoldt, B4hand, Barberio, Barnacle157, Beland, Bender235, Bentogoa, Bernard François, Betterworld, Bezenek, Bhavin, Biot, Bloodshedder, Bicomp, Branko, Breno, Brian.fsm, Brion VIBBER, Cam, Canthusus, CaptainVindaloo, Carmildo, Casey Abell, Cate, Cburnett, Cf. Hay, Chadernook, Cheesycow5, Ktatz, Clark42, Coasting, Conversion script, Coolcaesar, CrinklyCrunk, Ctm314, Cybercobra, Ctm314, Cynthia Rhoads, DARTH SIDIOUS 2, Damian Yerrick, Daniel Staal, DanielCD, Darkhalfact, DavidDW, DavidDouthitt, Denisarona, DerekLaw, Dgtsyb, Dicklyon, Disavian, Dmeranda, Dnas, Dogcow, Donjrude, Doradus, Dorgan65, Doug Bell, Drphilharmonic, Duffman, EagleOne, Ed g2s, Edmilne, Edward, Edwardando, Eekster, Ekashp, Electron9, Ellywa, Elwood j blues, EnOreg, EncMstr, Enjoia4586, Epr123, Eptin, Equendil, Erycl234, Erik Sandberg, Ethanhej, Etl, Evil Monkey, Evil saltine, Expensivehat, Falcon9x5, Falcor84, 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, Hasty001, Hcbekrowitz, Headbomb, Helix84, Here, Hoary, Holylampposts, Hpnguyen83, Hyad, IMSoP, Ilario, Imcdnzl, Imran, Indeterminate, Indinkgo, Inhumadecency, Inomyabcs, Intgr, Itai, J.delanoy, JTN, Jackqut7, James Mohr, JamesWatson, Jantangring, Jatkins, JesterXXV, Jimp, Jmdavid1789, Jnc, Joanjoq, John Vandenberg, Johnblade, Johnuniq, JonHarder, Jorunn, Jrogern, Jsoon eu, Jusdafax, JustAGal, KYPark, Kaare, Kasperd, Katieh5584, Kbrose, Kim Bruning, Kim Rubin, KnowledgeOfSelf, Kocio, Konman72, Koyaanis Qatsi, Krauss, Krellis, Kungming2, Kusma, Kvgng, Kyng, Labongo, Larree, Law, Layer, Leapfrog314, Lee Carre, Locketine, Logictheo, Lova Falk, Luna Santin, Magioladitis, Magister Mathematicae, Maltest, Mandarax, Mange01, Manop, Marcika, Martyman, Martyv, Matt Dunn, Matbrundage, Matthew Woodcraft, Matusz, Mav, McKoss, Mechanical digger, Meiskam, Mendel, Merlissimo, Metaclassing, Michael Hardy, Miles, MilesMi, Mintguy, Mothmolewa, Mrzaius, Mukkakukaku, Mwarren us, Mzje, NMChico24, Nasz, Navedahmed123, NawlinWiki,

Nealcardwell, Nealmcb, NewEnglandYankee, Ngriffeth, Nhorton, Nick C, Niteowlneils, Nivix, Nixdorf, Nknight, Nmacu, Nobody Ent, 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, Rbhagat0, Reaper Eternal, RedWolf, Reliabilesources, 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, Sitush, Sjakkalle, Skullketon, Smartse, Smig, Snaxe920, SnoFox, Spmion, Stahla92, StanQuayle, Staszek Lem, Stefan Milosevski, Stephan Leeds, Stephenb, Suffusion of Yellow, Sully, Sunray, SuperWiki, Suruena, Svick, Swwhitehead, Swpb, Ta bu shi da yu, Tagishsimon, Tarquin, Tassedethe, Techmonk, Techpro30, Template namespace initialisation script, Tfi, That Guy, From That Show!, Thatguyflint, The Anome, The Nut, TheOtherJesse, Theresa knott, Thingg, Thumperward, Thunderboltz, Thw1309, Tide rolls, Tim Watson, Timwi, TinaSDCE, Tmaufer, Tmchck, Tobias Hoevekamp, TomPhil, Topbanana, Tr606, Tyler, Typhoon, Ukexpat, Unyoyega, Vanis314, Vegaswikian, Victor Liu, Violetriga, W163, Wadamja, Waggars, Wavelength, Weregeek, West.andrew.g, Weylinp, Whereizben, Wiki104, Wikid77, Wikiklsc, William Avery, Wimt, Winston Chuen-Shih Yang, Wolfkeeper, Wonderstruck, Woohookitty, Wrs1864, XJamRastafire, Xeesh, Xojo, Xosé, Yakudza, Yas, Ydalal, Yudiweb, Yunshui, ZNott, Zac439, Zeerak88, Zfr, Zigger, Zoicon5, Zondor, Zundark, Zvezda1111, ^demon, شات صوتي, 831 anonymous edits



# Image Sources, Licenses and Contributors

**Image:Ethernet RJ45 connector p1160054.jpg** *Source:* [http://en.wikipedia.org/w/index.php?title=File:Ethernet\\_RJ45\\_connector\\_p1160054.jpg](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, Gnosygnu, Husky, Iamunknown, Mirithing, Myself488, Nethac DIU, Omegatron, Rocket000, Shanmugamp7, The Evil IP address, Wouterhagens, 22 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](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](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](http://en.wikipedia.org/w/index.php?title=File:Coreswitch_(2634205113).jpg) *License:* Creative Commons Attribution-Sharealike 2.0 *Contributors:* Dave Fischer

**File:Tcp state diagram fixed.svg** *Source:* [http://en.wikipedia.org/w/index.php?title=File:Tcp\\_state\\_diagram\\_fixed.svg](http://en.wikipedia.org/w/index.php?title=File:Tcp_state_diagram_fixed.svg) *License:* GNU Free Documentation License *Contributors:* Tcp\_state\_diagram\_new.svg: \*derivative work: Sergiodc2 (talk) Tcp\_state\_diagram.svg: dnet derivative work: Marty Pauley (talk)

**File:TCP CLOSE.svg** *Source:* [http://en.wikipedia.org/w/index.php?title=File:TCP\\_CLOSE.svg](http://en.wikipedia.org/w/index.php?title=File:TCP_CLOSE.svg) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Iustitia, 1 anonymous edits

**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

**File:OSI-model-Communication.svg** *Source:* <http://en.wikipedia.org/w/index.php?title=File:OSI-model-Communication.svg> *License:* Public Domain *Contributors:* Runtux

**File:SRI First Internetworked Connection diagram.jpg** *Source:* [http://en.wikipedia.org/w/index.php?title=File:SRI\\_First\\_Internetworked\\_Connection\\_diagram.jpg](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](http://en.wikipedia.org/w/index.php?title=File:SRI_Packet_Radio_Van.jpg) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Russavia

**Image:IP stack connections.svg** *Source:* [http://en.wikipedia.org/w/index.php?title=File:IP\\_stack\\_connections.svg](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](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

# License

---

Creative Commons Attribution-Share Alike 3.0 Unported  
[//creativecommons.org/licenses/by-sa/3.0/](https://creativecommons.org/licenses/by-sa/3.0/)

---