Click to prove you're human

Network protocols are often named in technical articles, but what are they? Network is easily understandable on its own, protocol as well, but together? What do we mean with “network protocol”? The ambiguous but accurate answer would be: what allowed you to read this article. I know, not very helpful. But it is the absolute truth. Network protocols are like languages: they exist to allow communication between different devices and services. Like human languages, you need to have a common language between you and your interlocutor, or you cannot communicate. There are network protocols for various tasks. All of them allow devices and services to talk to each other, but not all share the same type of content. Since the very first phone line was born, network protocols for sharing files, for printing from a remote printer, or for voice talking have been developed. Literally dozens of network protocols exist nowadays, from the basic and general ones to super-specific ones. How does a network protocol work? A network protocol is also called a “communication protocol”, or “network communication protocol”. Technically, a protocol of this class is defined as a set of rules for exchanging information over a network. It is similar to a human language, where the rules are the syntax and grammar of the language, while the info translates to the words we say or write. Any network protocol works according to certain rules, for a precise scope. The rules deal with “how the info should be transmitted”. They usually establish the order and meaning of some metadata that will tell the receiving device what protocol is being used. Similar to a human conversation, where somebody thanks you, and you know that you should politely reply with “you’re welcome”, network protocols have a set number of expected responses to each type of message. Not every device supports every protocol, like not every human speaks all languages. For instance, a device may support a network monitoring protocol, while an administrator may need to monitor its functioning. A different device may support a network management protocol because it is meant to manage other devices through one of those protocols. Unlike human languages that can be all used to speak and write, network protocols are more specific. There are network monitoring protocols that exchange info about the status of a device, to monitor it. Network security protocols securely exchange messages that contain sensitive info (by encrypting them via complicated mathematical algorithms). Network protocols exist to exchange files (FTP), show web pages (HTTP), connect to a remote device (SSH), monitor and configure it (SNMP), and so on. Some protocols have multiple functions. HTTP, for example, was born to distribute hypermedia texts (commonly called web pages) but has grown into multiple uses, ranging from streaming videos to voice calls. If you’re confused now, don’t worry, the experts felt the same way. Already in the 1970s, when most network systems we use today were being created, the OSI Network Model was developed to facilitate the interoperability between networks by following a system of conceptual layers. There are 7 layers in total, starting from the bottom, the hardware, and ending on the one closest to humans, the application layer. We will match each protocol to the various OSI model layers in the course of this article, when appropriate. The full list of layers, with an explanation, can be found here. Who uses network protocols? I guess you know the answer to this question already: everybody. Network protocols aren’t just for IT specialists and network administrators, since every one of us leverages them in one way or another. Simply browsing the internet, sending an email, receiving an SMS message, or chatting with friends means using a network protocol. Often even more than one at the same time. Experts have protocols for network monitoring to help them troubleshoot issues in a network, as it’s part of their job. Most people on the other hand won’t use any network monitoring protocols at all. Like natural languages, you learn and use only those that are useful to you. No need to know them all. But if you are more than an average user, you may want to know which protocols you can use to monitor a network or what network management protocols are out there. What types of network protocols exist? There are thousands of network protocols in use today, but we will focus on three main areas: network communication, network management, and network security. The reason is simple: the first are the most used, and by a great deal. The management ones are what we are more interested in and, last but certainly not least, the security protocols are of primary importance in monitoring. Network communication protocols Network communication protocols are the basic type of network protocols, born to transmit info over a network. Thus, they communicate info, be it generic or specific. Most of these protocols are commonly known and the base on which the whole Internet relies. IP and TCP/UDP The IP protocol is the root of the Internet. Without it, it wouldn’t have existed in the first place. How it works is easily understood, as its terminology has become part of what we all use to describe the Web. IP works by delivering packets from a source host to a destination IP address. It is also a routing protocol, as it established the concept of numerical addresses and their various classes. Depending on how the numbers are distributed, a packet can be meant for a local network (LAN) or an external one (WAN). Basically, the Internet is everybody’s WAN. IP is a connectionless protocol, meaning that there’s no established procedure to start communicating with a host. You just start talking and go on according to the reply (or lack thereof). Connection-oriented protocols instead have a rigid way to start communicating, involving a series of queries and replies that have to be followed for the info to be correctly transmitted and received. An example, built on IP, is TCP (Transmission Control Protocol). You may have heard of TCP/IP. It is so popular that it is being associated with the main network communication protocol of the internet. The other side of TCP is UDP (User Datagram Protocol). Unlike TCP, UDP is way less rigid, just taking care of how to package messages and sending them as fast as possible. Even though it is generally preferred over TCP for pure speed, you have no guarantee that the other end is correctly receiving your messages, nor actually understanding them. Among the OSI model layers, IP sits at the number 3, the network. TCP and UDP are above it, at layer 4, transport. DNS DNS is another key protocol of the internet. If IP gives every device an address (like a name of a person), DNS (Domain Name System) translates them to a specific position on a network (like the address of your home). DNS is the address registry of the Web, knowing which IP address corresponds to what domain name, and vice versa. DNS allows you to type a website in the browser and be led to the right one. Internet providers keep a “name server” for their customers that has records of all the IP addresses in their “zone”. The whole internet is divided into lots of these small zones. When you look for a website by its domain, your request moves from the one physically closest to you to the farthest necessary to have a positive answer. Once you have the IP address, the task of DNS is done, and you can move to other network protocols to start sharing the data you wanted. Perhaps unexpectedly, DNS is at the very top of the OSI layers, the application one, number 7. HTTP and FTP We are grouping these together because they have a similar scope. HTTP is usually the next protocol you go to once DNS gives you the IP address matching the domain you sought. HTTP takes care of transmitting the content that resides at that domain. Plain and simple. FTP does roughly the same. It stands for File Transfer Protocol and, as guessable from the name, focuses on transmitting files instead of webpages. Files can be extremely large, gigabytes and more. Once very popular, it has become less so in recent years as HTTP has taken over most of what FTP once did. Both HTTP and FTP work over TCP/UDP packets, and are at the application layer of the OSI model. SMTP An example of a protocol that is specific for one task only, SMTP (Simple Mail Transfer Protocol) transmits e-mails. And only that. It does not receive e-mails, since other protocols do that instead. SMTP only sends them. One task only, but done so well that it has been the network protocol for it since the early 1980s. SMTP not only sends your e-mails but adds the necessary headers (the "From" and "To" fields in e-mails are headers), and makes sure that the recipient has received the mail. Similarly to TCP, there are different requests to ensure proper sending and receiving of messages. SMTP is another protocol of the application layer of the OSI model. Network management protocols Over the years, the need to manage devices has grown alongside the size of networks. What was once a handful of computers, all in the same room, has become tens or hundreds of them, scattered in multiple locations worldwide, often under more than one organization. Managing them all is a difficult task. Network management protocols were developed to specifically ease that burden. SNMP As we have talked about Simple Network Management Protocol (SNMP) before, a long explanation is unnecessary. Suffice to say, SNMP is the granddad of network management protocols, hailing from the early 1980s and still largely used today. Many alternatives exist, though, which we discuss elsewhere. Its function is twofold: it not only manages (configures) remote devices, but can be used for network monitoring as well. The protocol is based on the concept of a manager and an agent. The first manages the various devices on which the agents are installed and keeps track of all collected metrics. Agents keep an eye on the device they are on and may send an alert to the manager if something is off. These are the basics of a fairly complex protocol. SNMP has been the main network monitoring protocol for over 4 decades. It is present on the OSI model at layer 7, application. Network security protocols In this section, we will see a few network protocols that focus on ensuring secure communication. They are not like human languages but more like a code that is preemptively exchanged between two people, where a symbol or number means something only to those who know the code. Network security protocols are the equivalent of such codes in computers. SSL and TLS SSL (Secure Sockets Layer) is indeed a layer over other protocols. Commonly, it is used with HTTP, making it HTTPS (S for secure). SSL was born in 1995 to provide a way to securely transmit info in the Netscape browser. Since then, all browsers have implemented it. How does it work though? It encrypts all data transmitted by using a combination of cryptographic keys and public certificates, along with a hashing algorithm. Lots of new words that can be easily translated: two computers agree, automatically, on how to encrypt the data they are going to exchange. The server provides its digital certificate, a file containing a long alphanumeric string representing the identity of the server. The client then knows that the server is who it claims to be, as only that server should have that specific certificate. After that, a couple of keys (other alphanumeric files) are generated based on the certificate and used to mathematically encrypt all the data transferred from then on. Sounds like a lot of complicated steps, but everything is transparent to the user. No step is manual, all is automatic. SSL was plagued with flaws and security issues from the beginning. It is now deprecated in favor of a more robust network security protocol, TLS (Transport Layer Security). It does exactly the same as SSL, just more safely. Curiously, neither SSL nor TLS fit clearly in the OSI model layers. They are supposed to be somewhere between the fourth and fifth, the transport and session layers. SSH Along the same years as SSL, the network security protocol SSH (Secure Shell Protocol) was also developed. SSH works in concept similar to SSL/TLS: a client and a server exchange a set of keys and/or certificates to encrypt the data they’re exchanging. We said “or” as with SSH it is not necessary to have a digital certificate. Authentication can be done via the typical pair of username and password, and then only digital keys are used. The certificate under SSH is useful to automatically authenticate, without a user present to type the name and password. SSH is thus a bit more flexible than SSL/TLS. It is usually not solely used as a network security protocol, but also as a protocol for monitoring network health. It can be considered as one of the network monitoring protocols as well. This is because SSH gives complete access to a remote computer, not just exchanging encrypted data. Once access is granted, you can manage, configure, and monitor the remote computer like it was yours. SSH is a great example of a network protocol that has more than one use. SSH is on the OSI model at layer 7, application. IPSec IPSec stands for Internet Protocol Security and the name should already tell you where it sits in the OSI model layer. It is the equivalent to IP, but secure. Thus, it is very low, at layer 3. You probably have not heard of IPSec per se, but of its most famous use: VPN. A virtual private network is like having remote computers like they were in your network, close to you. That works because IPSec takes care of creating a “tunnel”, an encrypted communication between computers, joining them virtually together as if they were physically close. The tunnel is established similarly to how SSH and SSL work, so no big surprises here. However, it is not used to exchange data with random servers, like SSL/TLS, or to access them, like SSH, but to have secure communication with multiple servers. That’s the definition of a VPN. IPSec is the network communication and security protocol that allows a VPN to work. It is typically used by companies to protect the communication among all the computers in their organization, even if they can’t physically be on the same network. What is a network monitoring protocol? Another type of network protocols that focus on one task are network monitoring protocols. They are subtypes of network communication protocols because, after all, they also exchange info. But whereas those are generic, network monitoring protocols are all about transmitting data that can be used by a network administrator to identify issues in the network. Compared with a human language, they are like people who regularly communicate with their doctors how they feel, and the doctor interprets their words to come up with a cure, if needed. Which protocol can be used for network monitoring? SNMP is probably the main one. It is often a good answer to the which protocol can be used for network monitor question. However, it is not the only one. A handful of other network monitoring protocols are a worthy choice as well. NetFlow and IPFIX Cisco’s proprietary NetFlow was the main competitor to SNMP in network monitoring for years. It is now superseded by IPFIX (IP Flow Information Export), which is based on it. Both protocols are “focused” network communication protocols as well. Focused, because instead of collecting everything that is being transmitted between network devices, they only collect parts of it. This way, they have a general idea of where traffic is coming from and going to. These parts are called “flows”. Compared to human languages, these flows are like talking in broad, general concepts and ideas instead of going into detail. For network monitoring, it can be advantageous to only have a general idea of how traffic moves. It eases the troubleshooting of issues and bottlenecks, initially. It is always possible to delve deeper in the info NetFlow and IPFIX captured as flows later, if desired. WMI and MI A popular choice for network monitoring under Windows is WMI (Windows Management Instrumentation). A fairly complex network monitoring protocol, WMI is similar in scope to a hybrid of SSH and SNMP. It allows both to access metrics to monitor a device and to have full access to it. Thus, powerful. It doesn’t have the security aspects of SSH, though, so it is not a network security protocol. WMI has morphed into MI (Management Infrastructure), supported by all the recent Windows’ versions. It works very similar to WMI but addresses some issues the original Windows-based network monitoring protocol had. ICMP A barebone network monitoring protocol is ICMP. An acronym for Internet Control Message Protocol, it is one of the oldest and simplest network protocols. It sits at a low OSI model layer, number 3, along with IP, and it hails from the same technological era. Its simplicity is its strength. All ICMP takes care of is checking that a device is up and replying. It is like two people repeatedly checking on each other with “are you there?”. The answer, in the case of ICMP, is either yes or no answer at all. In this case, an issue may be present and may be worth investigating further. It is nothing fancy, but for a first step in troubleshooting possible network problems, ICMP works just fine. IT monitoring platform The heart of the Checkmk platform is the High Performance Core, designed to scale up to millions of services monitored while still retaining a small footprint. Its REST API and many automations, such as the auto-registration of hosts, take manual work off your IT team’s shoulders. Checkmk not only monitors everything that powers your business, but also keeps it secure, thanks to granular access control, encryption, and 2FA. Acquire Monitor everything, don’t miss anything: our industry-leading selection of 2,000+ vendor-maintained plug-ins allows automated acquisition of health statuses and metrics from a multitude of data sources across servers, networks, clouds, containers, and applications. Don’t waste your time deciding what to monitor and how to configure it. Checkmk is ready in minutes, recognizes automatically which data is relevant to monitor, and displays key metrics in ready-to-use graphs. Analyze Benefit from pre-defined thresholds for alerts that provide out-of-the-box health assessment capabilities for the vast majority of hosts in your IT infrastructure. Analyze performance using built-in time-series analyzes, or compare metrics across multiple graphs at a glance. Review historical stats to forecast your future IT infrastructure requirements, or to determine the root cause of degraded performance and abnormal behavior. Alert Avoid notification overload by alerting only the team responsible for a specific point of failure. Easily configure alerts in granular manner and automate escalation of issues when they are not handled on time. Streamline your team’s workload by automatically creating tickets in third-party tools, such as ServiceNow, Jira, and many more, or send traditional text notifications via email, SMS or Slack. Resolve Get ultimate visibility from both built-in and custom dashboards packing key metrics and real-time insights. Empower your strategy with Business Intelligence, which provides a bird’s-eye view on the availability and performance of key business processes, by mapping application dependencies and/or aggregating various services and hosts into a single state. Take advantage of custom self-healing: use the alert handler to automatically trigger actions upon detection of new problems. Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. A network protocol is an established set of rules that determine how data is transmitted between different devices in the same network. Essentially, it allows connected devices to communicate with each other, regardless of any differences in their internal processes, structure or design. Network protocols are the reason you can easily communicate with people all over the world, and thus play a critical role in modern digital communications. Similar to the way that speaking the same language simplifies communication between two people, network protocols make it possible for devices to interact with each other because of predetermined rules built into devices’ software and hardware. Neither local area networks (LAN) nor wide area networks (WAN) could function the way they do today without the use of network protocols. Network protocols take large-scale processes and break them down into small, specific tasks or functions. This occurs at every level of the network, and each function must cooperate at each level to complete the larger task at hand. The term protocol suite refers to a set of smaller network protocols working in conjunction with each other.Network protocols are typically created according to industry standard by various networking or information technology organizations. The following groups have defined and published different network protocols:While network protocol models generally work in similar ways, each protocol is unique and operates in the specific way detailed by the organization that created it.Who Uses Network Protocols?Network protocols aren’t only relevant to certified network specialists or IT professionals. Billions of people use network protocols daily, whether they know it or not. Every time you use the internet, you leverage network protocols. Though you may not know how network protocols work or how frequently you encounter them, they are necessary for using the internet or digital communications in any capacity. List of Network ProtocolsThere are thousands of different network protocols, but they all perform one of three primary actions: Communication Network managementSecurityEach type is necessary to use network devices swiftly and safely, and they work together to facilitate that usage. Communication Communication protocols allow different network devices to communicate with each other. They are used in both analog and digital communications and can be used for important processes, ranging from transferring files between devices to accessing the internet. Common types of communication protocols include the following:Automation: These protocols are used to automate different processes in both commercial and personal settings, such as in smart buildings, cloud technology or self-driving vehicles.Instant messaging: Instantaneous, text-based communications on smartphones and computers occur because of a number of different instant messaging network protocols.Routing: Routing protocols permit communication between routers and other network devices. There are also routing protocols specifically for ad hoc networks.Bluetooth: Ever-popular Bluetooth devices — including headsets, smartphones and computers — work due to a variety of different Bluetooth protocols.File transfer: If you have ever moved files from one device to another, either via a physical or digital medium, you’ve used file transfer protocols (FTP). Internet Protocol: Internet Protocol (IP) allows data to be sent between devices via the internet. The internet could not operate as it currently does without IP. Network ManagementNetwork management protocols define and describe the various procedures needed to effectively operate a computer network. These protocols affect various devices on a single network — including computers, routers and servers — to ensure each one, and the network as a whole, perform optimally. The functions of network management protocols include the following:Connection: These protocols establish and maintain stable connections between different devices on the same network.Link aggregation: Link aggregation protocols allow you to combine multiple network connections into one link between two devices. This works to increase the strength of the connection and helps sustain the connection should one of the links fail.Troubleshooting: Troubleshooting protocols allow network administrators to identify errors affecting the network, evaluate the quality of the network connection, and determine how administrators can fix any issues. SecuritySecurity protocols, also called cryptographic protocols, work to ensure that the network and the data sent over it are protected from unauthorized users. Common functions of security network protocols include the following:Encryption: Encryption protocols protect data and secure areas by requiring users to input a secret key or password in order to access that information.Entity Authentication: Entity authentication protocols create a system that requires different devices or users on a network to verify their identity before accessing secure areas.Transportation: Transportation security protocols protect data while it is transported from one network device to another. Network Protocol ExampleWhether you know it or not, you’ve absolutely encountered network protocols when using electronic devices – and some of them are readily identifiable. Here are a few examples of the most commonly used network protocols:Hypertext Transfer Protocol (HTTP): This Internet Protocol defines how data is transmitted over the internet and determines how web servers and browsers should respond to commands. This protocol (or its secure counterpart, HTTPS) appears at the beginning of various URLs or web addresses online.Secure Socket Shell (SSH): This protocol provides secure access to a computer, even if it’s on an unsecured network. SSH is particularly useful for network administrators who need to manage different systems remotely. Short Message Service (SMS): This communications protocol was created to send and receive text messages over cellular networks. SMS refers exclusively to text-based messages. Pictures, videos or other media require Multimedia Messaging Service (MMS), an extension of the SMS protocol.Network protocols do not simply define how devices and processes work; they define how devices and processes work together. Without these predetermined conventions and rules, the internet would lack the necessary infrastructure it needs to be functional and useable. Network protocols are the foundation of modern communications, without which the digital world could not stand.CompTIA Network+ covers computer networking topics including network protocols. Download the exam objectivesto see all the topics covered by this IT certification. Read more about Computer Networks. Nuestra experiencia siempre contigo. Protegemos tu negocio de cualquier amenaza. 4.88MDD, costo promedio global de una filtración de datos en 20242,244Ataques diarios enfrenta un solo ordenador, uno cada 39 segundos.277Días toma en promedio identificar y contener una filtración activa System for exchanging messages between computing systems A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any variation of a physical quantity. The protocol defines the rules, syntax, semantics, and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.[1] Communicating systems use well-defined formats for exchanging various messages. Each message has an exact meaning intended to elicit a response from a range of possible responses predetermined for that particular situation. The specified behavior is typically independent of how it is to be implemented. Communication protocols have to be agreed upon by the parties involved.[2] To reach an agreement, a protocol may be developed into a technical standard. A programming language describes the same for computations, so there is a close analogy between protocols and programming languages: protocols are to communication what programming languages are to computations.[3] An alternate formulation states that protocols are to communication what algorithms are to computation.[4] Multiple protocols often describe different aspects of a single communication. A group of protocols designed to work together is known as a protocol suite; when implemented in software they are a protocol stack. Internet communication protocols are published by the Internet Engineering Task Force (IETF). The IEEE (Institute of Electrical and Electronics Engineers) handles wired and wireless networking and the International Organization for Standardization (ISO) handles other types. The ITU-T handles telecommunications protocols and formats for the public switched telephone network (PSTN). As the PSTN and Internet converge, the standards are also being driven towards convergence. Further information: Protocol Wars The first use of the term protocol in a modern data-commutation context occurs in April 1967 in a memorandum entitled A Protocol for Use in the NPL Data Communications Network. Under the direction of Donald Davies, who pioneered packet switching at the National Physical Laboratory in the United Kingdom, it was written by Roger Scantlebury and Keith Bartlett for the NPL network.[5][6][7][8][9] On the ARPANET, the starting point for host-to-host communication in 1969 was the 1822 protocol, written by Bob Kahn, which defined the transmission of messages to an IMP.[10] The Network Control Program (NCP) for the ARPANET, developed by Steve Crocker and other graduate students including Jon Postel and Vint Cerf, was first implemented in 1970.[11] The NCP interface allowed application software to connect across the ARPANET by implementing higher-level communication protocols, an early example of the protocol layering concept.[12] The CYCLADES network, designed by Louis Pouzin in the early 1970s was the first to implement the end-to-end principle, and make the hosts responsible for the reliable delivery of data on a packet-switched network, rather than this being a service of the network itself.[13] His team was the first to tackle the highly complex problem of providing user applications with a reliable virtual circuit service while using a best-effort service, an early contribution to what will be the Transmission Control Protocol (TCP).[14][15][16] Bob Metcalfe and others at Xerox PARC outlined the idea of Ethernet and the PARC Universal Packet (PUP) for internetworking.[17] Research in the early 1970s by Bob Kahn and Vint Cerf led to the formulation of the Transmission Control Program (TCP).[18] Its RFC 675 specification was written by Cerf with Yogen Dalal and Carl Sunshine in December 1974, still a monolithic design at this time. The International Network Working Group agreed on a connectionless datagram standard which was presented to the CCITT in 1975 but was not adopted by the CCITT nor by the ARPANET.[19] Separate international research, particularly the work of Rémi Després, contributed to the development of the X.25 standard, based on virtual circuits, which was adopted by the CCITT in 1976.[20][21] Computer manufacturers developed proprietary protocols such as IBM's Systems Network Architecture (SNA), Digital Equipment Corporation's DECnet and Xerox Network Systems.[22] TCP software was redesigned as a modular protocol stack, referred to as TCP/IP. This was installed on SATNET in 1982 and on the ARPANET in January 1983. The development of a complete Internet protocol suite by 1989, as outlined in RFC 1122 and RFC 1123, laid the foundation for the growth of TCP/IP as a comprehensive protocol suite as the core component of the emerging Internet.[23] International work on a reference model for communication standards led to the OSI model, published in 1984. For a period in the late 1980s and early 1990s, engineers, organizations and nations became polarized over the issue of which standard, the OSI model or the Internet protocol suite, would result in the best and most robust computer networks.[24][25][26] The information exchanged between devices through a network or other media is governed by rules and conventions that can be set out in communication protocol specifications. The nature of communication, the actual data exchanged and any state-dependent behaviors, is defined by these specifications. In digital computing systems, the rules can be expressed by algorithms and data structures. Protocols are to communication what algorithms or programming languages are to computations.[3][4] Operating systems usually contain a set of cooperating processes that manipulate shared data to communicate with each other. This communication is governed by well-understood protocols, which can be embedded in the process code itself.[27][28] In contrast, because there is no shared memory, communicating systems have to communicate with each other using a shared transmission medium. Transmission is not necessarily reliable, and individual systems may use different hardware or operating systems. To implement a networking protocol, the protocol software modules are interfaced with a framework implemented on the machine's operating system. This framework implements the networking functionality of the operating system.[29] When protocol algorithms are expressed in a portable programming language the protocol software may be made operating system independent. The best-known frameworks are the TCP/IP model and the OSI model. At the time the Internet was developed, abstraction layering had proven to be a successful design approach for both compiler and operating system design and, given the similarities between programming languages and communication protocols, the originally monolithic networking programs were decomposed into cooperating protocols.[30] This gave rise to the concept of layered protocols which nowadays forms the basis of protocol design.[31] Systems typically do not use a single protocol to handle a transmission. Instead they use a set of cooperating protocols, sometimes called a protocol suite.[32] Some of the best-known protocol suites are TCP/IP, IPX/SPX, X.25, AX.25 and AppleTalk. The protocols can be arranged based on functionality in groups, for instance, there is a group of transport protocols. The functionalities are mapped onto the layers, each layer solving a distinct class of problems relating to, for instance: application-, transport-, internet- and network interface-functions.[33] To transmit a message, a protocol has to be selected from each layer. The selection of the next protocol is accomplished by extending the message with a protocol selector for each layer.[34] There are two types of communication protocols, based on their representation of the content being carried: text-based and binary.[35] A text-based protocol or plain text protocol represents its content in human-readable format, often in plain text encoded in a machine-readable encoding such as ASCII or UTF-8, or in structured text-based formats such as Intel hex format, XML or JSON. The immediate human readability stands in contrast to native binary protocols which have inherent benefits for use in a computer environment (such as ease of mechanical parsing and improved bandwidth utilization). Network applications have various methods of encapsulating data. One method very common with Internet protocols is a text oriented representation that transmits requests and responses as lines of ASCII text, terminated by a newline character (and usually a carriage return character). Examples of protocols that use plain, human-readable text for its commands are FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), early versions of HTTP (Hypertext Transfer Protocol), and the finger protocol.[36] Text-based protocols are typically optimized for human parsing and interpretation and are therefore suitable whenever human inspection of protocol contents is required, such as during debugging and during early protocol development design phases. A binary protocol utilizes all values of a byte, as opposed to a text-based protocol which only uses values corresponding to human-readable characters in ASCII encoding. Binary protocols are intended to be read by a machine rather than a human being. Binary protocols have the advantage of terseness, which translates into speed of transmission and interpretation.[37] Binary have been used in the normative documents describing modern standards like EbXML, HTTP/2, HTTP/3 and EDOC.[38] An interface in UML[39] may also be considered a binary protocol. Getting the data across a network is only part of the problem for a protocol. The data received has to be evaluated in the context of the progress of the conversation, so a protocol must include rules describing the context. These kinds of rules are said to express the syntax of the communication. Other rules determine whether the data is meaningful for the context in which the exchange takes place. These kinds of rules are said to express the semantics of the communication. Messages are sent and received on communicating systems to establish communication. Protocols should therefore specify rules governing the transmission. In general, much of the following should be addressed:[40] Data formats for data exchange Digital message bitstrings are exchanged. The bitstrings are divided in fields and each field carries information relevant to the protocol. Conceptually the bitstring is divided into two parts called the header and the payload. The actual message is carried in the payload. The header area contains the fields with relevance to the operation of the protocol. Bitstrings longer than the maximum transmission unit (MTU) are divided in pieces of appropriate size.[41] Address formats for data exchange Addresses are used to identify both the sender and the intended receiver(s). The addresses are carried in the header area of the bitstrings, allowing the receivers to determine whether the bitstrings are of interest and should be processed or should be ignored. A connection between a sender and a receiver can be identified using an address pair (sender address, receiver address). Usually, some address values have special meanings. An all-1s address could be taken to mean an addressing of all stations on the network, so sending to this address would result in a broadcast on the local network. The rules describing the meanings of the address value are collectively called an addressing scheme.[42] Address mapping Sometimes protocols need to map addresses of one scheme on addresses of another scheme. For instance, to translate a logical IP address specified by the application to an Ethernet MAC address. This is referred to as address mapping.[43] Routing When systems are not directly connected, intermediary systems along the route to the intended receiver(s) need to forward messages on behalf of the sender. On the Internet, the networks are connected using routers. The interconnection of networks through routers is called internetworking. Detection of transmission errors Error detection is necessary on networks where data corruption is possible. In a common approach, a CRC of the data area is added to the end of packets, making it possible for the receiver to detect differences caused by corruption. The receiver rejects the packets on CRC differences and arranges somehow for retransmission.[44] Acknowledgements Acknowledgement of correct reception of packets is required for connection-oriented communication. Acknowledgments are sent from receivers back to their respective senders.[45] Loss of information - timeouts and retries Packets may be lost on the network or be delayed in transit. To cope with this, under some protocols, a sender may expect an acknowledgment of correct reception from the receiver within a certain amount of time. Thus, on timeouts, the sender may need to retransmit the information.[a] In case of a permanently broken link, the retransmission has no effect, so the number of retransmissions is limited. Exceeding the retry limit is considered an error.[46] Direction of information flow Direction needs to be addressed if transmissions can only occur in one direction at a time as on half-duplex links or from one sender at a time as on a shared medium. This is known as media access control. Arrangements have to be made to accommodate the case of collision or contention where two parties respectively simultaneously transmit or wish to transmit.[47] Sequence control If long bitstrings are divided into pieces and then sent on the network individually, the pieces may get lost or delayed or, on some types of networks, take different routes to their destination. As a result, pieces may arrive out of sequence. Retransmissions can result in duplicate pieces. By marking the pieces with sequence information at the sender, the receiver can determine what was lost or duplicated, ask for necessary retransmissions and reassemble the original message.[48] Flow control Flow control is needed when the sender transmits faster than the receiver or intermediate network equipment can process the transmissions. Flow control can be implemented by messaging from receiver to sender.[49] Queueing Communicating processes or state machines employ queues (or "buffers"), usually FIFO queues, to deal with the messages in the order sent, and may sometimes have multiple queues with different prioritization. Systems engineering principles have been applied to create a set of common network protocol design principles. The design of complex protocols often involves decomposition into simpler, cooperating protocols. Such a set of cooperating protocols is sometimes called a protocol family or a protocol suite,[32] within a conceptual framework. Communicating systems operate concurrently. An important aspect of concurrent programming is the synchronization of software for receiving and transmitting messages of communication in proper sequencing. Concurrent programming has traditionally been a topic in operating systems theory texts.[50] Formal verification seems indispensable because concurrent programs are notorious for the hidden and sophisticated bugs they contain.[51] A mathematical approach to the study of concurrency and communication is referred to as communicating sequential processes (CSP).[52] Concurrency can also be modeled using finite-state machines, such as Mealy and Moore machines. Mealy and Moore machines are in use as design tools in digital electronics systems encountered in the form of hardware used in telecommunication or electronic devices in general.[53] [better source needed] The literature presents numerous analogies between computer communication and programming. In analogy, a transfer mechanism of a protocol is comparable to a central processing unit (CPU). The framework introduces rules that allow the programmer to design cooperating protocols independently of one another. The TCP/IP model or Internet layering scheme and its relation to some common protocols. In modern protocol design, protocols are layered to form a protocol stack. Layering is a design principle that divides the protocol design task into smaller steps, each of which accomplishes a specific part, interacting with the other parts of the protocol only in a small number of well-defined ways. Layering allows the parts of a protocol to be designed and tested without a combinatorial explosion of cases, keeping each design relatively simple. The communication protocols in use on the Internet are designed to function in diverse and complex settings. Internet protocols are designed for simplicity and modularity and fit into a coarse hierarchy of functional layers defined in the Internet Protocol Suite.[54] The first two cooperating protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP) resulted from the decomposition of the original Transmission Control Program, a monolithic communication protocol, into this layered communication suite. The OSI model was developed internationally based on experience with networks that predated the internet as a reference model for general communication with much stricter rules of protocol interaction and rigorous layering. Typically, application software is built upon a robust data transport layer. Underlying this transport layer is a datagram delivery and routing mechanism that is typically connectionless in the Internet. Packet relaying across networks happens over another layer that involves only network link technologies, which are often specific to certain physical layer technologies, such as Ethernet. Layering provides opportunities to exchange technologies when needed, for example, protocols are often stacked in a tunneling arrangement to accommodate the connection of dissimilar networks. For example, IP may be tunneled across an Asynchronous Transfer Mode (ATM) network. Figure 3. Message flows using a protocol suite. Black loops show the actual messaging loops, red loops are the effective communication between layers enabled by the lower layers. Protocol layering forms the basis of protocol design.[31] It allows the decomposition of single, complex protocols into simpler, cooperating protocols.[54] The protocol layers each solve a distinct class of communication problems. Together, the layers make up a layering scheme or model. Computations deal with algorithms and data; Communication involves protocols and messages; So the analog of a data flow diagram is some kind of message flow diagram.[4] To visualize protocol layering and protocol suites, a diagram of the message flows in and between two systems, A and B, is shown in figure 3. The systems, A and B, both make use of the same protocol suite. The vertical flows (and protocols) are in-system and the horizontal message flows (and protocols) are between systems. The message flows are governed by rules, and data formats specified by protocols. The blue lines mark the boundaries of the (horizontal) protocol layers. Figure 5: Protocol and software layering. The software modules implementing the protocols are represented by cubes. The information flow between the modules is represented by arrows. The (top two horizontal) red arrows are virtual. The blue lines mark the layer boundaries. The software supporting protocols has a layered organization and its relationship with protocol layering is shown in figure 5. To send a message on system A, the top-layer software module interacts with the module directly below it and hands over the message to be encapsulated. The lower module fills in the header data in accordance with the protocol it implements and interacts with the bottom module which sends the message over the communications channel to the bottom module of system B. On the receiving system B the reverse happens, so ultimately the message gets delivered in its original form to the top module of system B.[55] Program translation is divided into subproblems. As a result, the translation software is layered as well, allowing the software layers to be designed independently. The same approach can be seen in the TCP/IP layering.[56] The modules below the application layer are generally considered part of the operating system. Passing data between these modules is much less expensive than passing data between an application program and the transport layer. The boundary between the application layer and the transport layer is called the operating system boundary.[57] Strictly adhering to a layered model, a practice known as strict layering, is not always the best approach to networking.[58] Strict layering can have a negative impact on the performance of an implementation.[59] Although the use of protocol layering is today ubiquitous across the field of computer networking, it has been historically criticized by many researchers[60] as abstracting the protocol stack in this way may cause a higher layer to duplicate the functionality of a lower layer, a prime example being error recovery on both a per-link basis and an end-to-end basis.[61] Commonly recurring problems in the design and implementation of communication protocols can be addressed by software design patterns.[62][63][64][65][66] Popular formal methods of describing communication syntax are Abstract Syntax Notation One (an ISO standard) and augmented Backus–Naur form (an IETF standard). Finite-state machine models are used to formally describe the possible interactions of the protocol.[67][68] and communicating finite-state machines[69] For communication to occur, protocols have to be selected. The rules can be expressed by algorithms and data structures. Hardware and operating system independence is enhanced by expressing the algorithms in a portable programming language. Source independence of the specification provides wider interoperability. Protocol standards are commonly created by obtaining the approval or support of a standards organization, which initiates the standardization process. The members of the standards organization agree to adhere to the work result on a voluntary basis. Often the members are in control of large market shares relevant to the protocol and in many cases, standards are enforced by law or the government because they are thought to serve an important public interest, so getting approval can be very important for the protocol. The need for protocol standards can be shown by looking at what happened to the Binary Synchronous Communications (BSC) protocol invented by IBM. BSC is an early link-level protocol used to connect two separate nodes. It was originally not intended to be used in a multinode network, but doing so revealed several deficiencies of the protocol. In the absence of standardization, manufacturers and organizations felt free to enhance the protocol, creating incompatible versions on their networks. In some cases, this was deliberately done to discourage users from using equipment from other manufacturers. There are more than 50 variants of the original bi-sync protocol. One can assume, that a standard would have prevented at least some of this from happening.[29] In some cases, protocols gain market dominance without going through a standardization process. Such protocols are referred to as de facto standards. De facto standards are common in emerging markets, niche markets, or markets that are monopolized (or oligopolized). They can hold a market in a very negative grip, especially when used to scare away competition. From a historical perspective, standardization should be seen as a measure to counteract the ill-effects of de facto standards. Positive exceptions exist; a de facto standard operating system like Linux does not have this negative grip on its market, because the sources are published and maintained in an open way, thus inviting competition. Some of the standards organizations of relevance for communication protocols are the International Organization for Standardization (ISO), the International Telecommunication Union (ITU), the Institute of Electrical and Electronics Engineers (IEEE), and the Internet Engineering Task Force (IETF). The IETF maintains the protocols in use on the Internet. The IEEE controls many software and hardware protocols in the electronics industry for commercial and consumer devices. The ITU is an umbrella organization of telecommunication engineers designing the public switched telephone network (PSTN), as well as many radio communication systems. For marine electronics the NMEA standards are used. The World Wide Web Consortium (W3C) produces protocols and standards for Web technologies. International standards organizations are supposed to be more impartial than local organizations with a national or commercial self-interest to consider. Standards organizations also do research and development for standards of the future. In practice, the standards organizations mentioned, cooperate closely with each other.[70] Multiple standards bodies may be involved in the development of a protocol. If they are uncoordinated, then the result may be multiple, incompatible definitions of a protocol, or multiple, incompatible interpretations of messages; important invariants in one definition (e.g., that time-to-live values are monotone decreasing to prevent stable routing loops) may not be respected in another.[71] In the ISO, the standardization process starts off with the commissioning of a sub-committee workgroup. The workgroup issues working drafts and discussion documents to interested parties (including other standards bodies) in order to provoke discussion and comments. This will generate a lot of questions, much discussion and usually some disagreement. These comments are taken into account and a draft proposal is produced by the working group. After feedback, modification, and compromise the proposal reaches the status of a draft international standard, and ultimately an international standard. International standards are reissued periodically to handle the deficiencies and reflect changing views on the subject.[72] OSI modelby layer 7. Application layer NNTP SIP SSI DNS FTP Gopher HTTP (HTTP/3) NFS NTP SMPP SSH SMTP SNMP Telnet DHCP NETCONF more.... 6. Presentation layer MIME XDR ASN.1 ASCII TLS PGP 5. Session layer Named pipe NetBIOS SAP PPTP RTP SOCKS X.225[73] 4. Transport layer TCP UDP SCTP DCCP QUIC SPX 3. Network layer IP IPv4 IPv6 ICMP (ICMPv6) IPsec IGMP IPX IS-IS AppleTalk X.25 PLP 2. Data link layer ATM ARP SDLC HDLC CSLIP SLIP GFP PLIP IEEE 802 LLC MAC L2TP Frame Relay ITU-T G.hn DLL PPP X.25 LAPB Q.922 LAPF 1. Physical layer RS-232 RS-449 ITU-T V-Series I.430 I.431 PDH SONET/SDH PON OTN DSL IEEE 802 IEEE 1394 ITU-T G.hn PHY USB Bluetooth X.21 vte A lesson learned from ARPANET, the predecessor of the Internet, was that protocols need a framework to operate. It is therefore important to develop a general-purpose, future-proof framework suitable for structured protocols (such as layered protocols) and their standardization. This would prevent protocol standards with overlapping functionality and would allow clear definition of the responsibilities of a protocol at the different levels (layers).[74] This gave rise to the Open Systems Interconnection model (OSI model), which is used as a framework for the design of standard protocols and services conforming to the various layer specifications.[75] In the OSI model, communicating systems are assumed to be connected by an underlying physical medium providing a basic transmission mechanism. The layers above it are numbered. Each layer provides service to the layer above it using the services of the layer immediately below it. The top layer provides services to the application process. The layers communicate with each other by means of an interface, called a service access point. Corresponding layers at each system are called peer entities. To communicate, two peer entities at a given layer use a protocol specific to that layer which is implemented by using services of the layer below.[76] For each layer, there are two types of standards: protocol standards defining how peer entities at a given layer communicate, and service standards defining how a given layer communicates with the layer above it. In the OSI model, the layers and their functionality are (from highest to lowest layer): The Application layer may provide the following services to the application processes: identification of the intended communication partners, establishment of the necessary authority to communicate, determination of availability and authentication of the partners, agreement on privacy mechanisms for the communication, agreement on responsibility for error recovery and procedures for ensuring data integrity, synchronization between cooperating application processes, identification of any constraints on syntax (e.g. character sets and data structures), determination of cost and acceptable quality of service, selection of the dialogue discipline, including required logon and logoff procedures.[77] The presentation layer may provide the following services to the application layer: a request for the establishment of a session, data transfer, negotiation of the syntax to be used between the application layers, any necessary syntax transformations, formatting and special purpose transformations (e.g., data compression and data encryption).[78] The session layer may provide the following services to the presentation layer: establishment and release of session connections, normal and expedited data exchange, a quarantine service which allows the sending presentation entity to instruct the receiving session entity not to release data to its presentation entity without permission, interaction management so presentation entities can control whose turn it is to perform certain control functions, resynchronization of a session connection, reporting of unrecoverable exceptions to the presentation entity.[79] The transport layer provides reliable and transparent data transfer in a cost-effective way as required by the selected quality of service. It may support the multiplexing of several transport connections on to one network connection or split one transport connection into several network connections.[80] The network layer does the setup, maintenance and release of network paths between transport peer entities. When relays are needed, routing and relay functions are provided by this layer. The quality of service is negotiated between network and transport entities at the time the connection is set up. This layer is also responsible for network congestion control.[81] The data link layer does the setup, maintenance and release of data link connections. Errors occurring in the physical layer are detected and may be corrected. Errors are reported to the network layer. The exchange of data link units (including flow control) is defined by this layer.[82] The physical layer describes details like the electrical characteristics of the physical connection, the transmission techniques used, and the setup, maintenance and clearing of physical connections.[83] In contrast to the TCP/IP layering scheme, which assumes a connectionless network, RM/OSI assumed a connection-oriented network.[84] Connection-oriented networks are more suitable for wide area networks and connectionless networks are more suitable for local area networks. Connection-oriented communication requires some form of session and (virtual) circuits, hence the (in the TCP/IP model lacking) session layer. The constituent members of ISO were mostly concerned with wide area networks, so the development of RM/OSI concentrated on connection-oriented networks and connectionless networks were first mentioned in an addendum to RM/OSI[85][86] and later incorporated into an update to RM/OSI.[87] At the time,[when?] the IETF had to cope with this and the fact that the Internet needed protocols that simply were not there.[citation needed] As a result, the IETF developed its own standardization process based on "rough consensus and running code".[88] The standardization process is described by RFC 2026. Nowadays, the IETF has become a standards organization for the protocols in use on the Internet. RM/OSI has extended its model to include connectionless services and because of this, both TCP and IP could be developed into international standards.[citation needed] Main article: Wire data The wire image of a protocol is the information that a non-participant observer is able to glean from observing the protocol messages, including both information explicitly given meaning by the protocol, but also inferences made by the observer.[89] Unencrypted protocol metadata is one source making up the wire image, and side-channels including packet timing also contribute.[90] Different observers with different vantages may see different wire images.[91] The wire image is relevant to end-user privacy and the extensibility of the protocol.[92] If some portion of the wire image is not cryptographically authenticated, it is subject to modification by intermediate parties (i.e., middleboxes), which can influence protocol operation.[90] Even if authenticated, if a portion is not encrypted, it will form part of the wire image, and intermediate parties may intervene depending on its content (e.g., dropping packets with particular flags). Signals deliberately intended for intermediary consumption may be left authenticated but unencrypted.[93] The wire image can be deliberately engineered, encrypting parts that intermediaries should not be able to observe and providing signals for what they should be able to.[94] If provided signals are decoupled from the protocol's operation, they may become untrustworthy.[95] Benign network management and research are affected by metadata encryption; protocol designers must balance observability for operability and research against ossification resistance and end-user privacy.[92] The IETF announced in 2014 that it had determined that large-scale surveillance of protocol operations is an attack due to the ability to infer information from the wire image about users and their behaviour,[96] and that the IETF would "work to mitigate pervasive monitoring" in its protocol designs;[97] this had not been done systematically previously.[97] The Internet Architecture Board recommended in 2023 that disclosure of information by a protocol to the network should be intentional,[98] performed with the agreement of both recipient and sender,[99] authenticated to the degree possible and necessary,[100] only acted upon to the degree of its trustworthiness,[101] and minimised and provided to a minimum number of entities.[102][103] Engineering the wire image and controlling what signals are provided to network elements was a "developing field" in 2023, according to the IAB.[104] Main article: protocol ossification Protocol ossification is the loss of flexibility, extensibility and evolvability of network protocols. This is largely due to middleboxes that are sensitive to the wire image of the protocol, and which can interrupt or interfere with messages that are valid but which the middlebox does not correctly recognize.[105] This is a violation of the end-to-end principle.[106] Secondary causes include inflexibility in endpoint implementations of protocols.[107] Ossification is a major issue in Internet protocol design and deployment, as it can prevent new protocols or extensions from being deployed on the Internet, or place strictures on the design of new protocols; new protocols may have to be encapsulated in an already-deployed protocol or mimic the wire image of another protocol.[108] Because of ossification, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the only practical choices for transport protocols on the Internet,[109] and TCP itself has significantly ossified, making extension or modification of the protocol difficult.[110] Recommended methods of preventing ossification include encrypting protocol metadata,[111] and ensuring that extension points are exercised and wire image variability is exhibited as fully as possible;[112] remedying existing ossification requires coordination across protocol participants.[113] QUIC is the first IETF transport protocol to have been designed with deliberate anti-ossification properties.[89] Classification schemes for protocols usually focus on the domain of use and function. As an example of domain of use, connection-oriented protocols and connectionless protocols are used on connection-oriented networks and connectionless networks respectively. An example of function is a tunneling protocol, which is used to encapsulate packets in a high-level protocol so that the packets can be passed across a transport system using the high-level protocol. A layering scheme combines both function and domain of use. The dominant layering schemes are the ones developed by the IETF and by ISO. Despite the fact that the

underlying assumptions of the layering schemes are different enough to warrant distinguishing the two, it is a common practice to compare the two by relating common protocols to the layers of the two schemes.[114] The layering scheme from the IETF is called Internet layering or TCP/IP layering. The layering scheme from ISO is called the OSI model or ISO layering. In networking equipment configuration, a term-of-art distinction is often drawn: The term protocol strictly refers to the transport layer, and the term service refers to protocols utilizing a protocol for transport. In the common case of TCP and UDP, services are distinguished by port numbers. Conformance to these port numbers is voluntary, so in content inspection systems the term service strictly refers to port numbers, and the term application is often used to refer to protocols identified through inspection signatures. Cryptographic protocol – Aspect of cryptography Lists of network protocols Protocol Builder – Programming tool to build network connectivity components ^ Failure to receive an acknowledgment indicates that either the original transmission or the acknowledgment was lost. The sender has no means to distinguish these cases and therefore, to ensure all data is received, must make the conservative assumption that the original transmission was lost. ^ US 7529565, Hilpisch, Robert E.; Duchscher, Rob & Seel, Mark et al., "Wireless communication protocol", published 5 May 2009, assigned to Starkey Laboratories Inc. and Oticon AS ^ Protocol, Encyclopædia Britannica, archived from the original on 12 September 2012, retrieved 24 September 2012 ^ a b Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, "They (protocols) are to communication what programming languages are to computation" ^ a b c Comer 2000, Sect. 1.3 - Internet Services, p. 3, "Protocols are to communication what algorithms are to computation" ^ Naughton, John (24 September 2015). A Brief History of the Future. Orion. ISBN 978-1-4746-0277-8. ^ Cambell-Kelly, Martin (1987). "Data Communications at the National Physical Laboratory (1965-1975)". Annals of the History of Computing. 9 (3/4): 221–247. doi:10.1109/MAHC.1987.10023. S2CID 8172150. ^ Pelkey, James L. "6.1 The Communications Subnet: BBN 1969". Entrepreneurial Capitalism and Innovation: A History of Computer Communications 1968–1988. As Kahn recalls: ... Paul Baran's contributions ... I also think Paul was motivated almost entirely by voice considerations. If you look at what he wrote, he was talking about switches that were low-cost electronics. The idea of putting powerful computers in these locations hadn't quite occurred to him as being cost effective. So the idea of computer switches was missing. The whole notion of protocols didn't exist at that time. And the idea of computer-to-computer communications was really a secondary concern. ^ Waldrop, M. Mitchell (2018). The Dream Machine. Stripe Press. p. 286. ISBN 978-1-953953-36-0. Baran had put more emphasis on digital voice communications than on computer communications. ^ Kleinrock, L. (1978). "Principles and lessons in packet communications". Proceedings of the IEEE. 66 (11): 1320–1329. doi:10.1109/PROC.1978.11143. ISSN 0018-9219. Paul Baran ... focused on the routing procedures and on the survivability of distributed communication systems in a hostile environment, but did not concentrate on the need for resource sharing in its form as we now understand it; indeed, the concept of a software switch was not present in his work. ^ Interface Message Processor: Specifications for the Interconnection of a Host and an IMP (PDF) (Report). Bolt Beranek and Newman (BBN). Report No. 1822. ^ BOOKS, HIGH DEFINITION. UGC -NET/JRF/SET PTP & Guide Teaching and Research Aptitude: UGC -NET By HD. High Definition Books. ^ "NCP – Network Control Program". Living Internet. Archived from the original on 7 August 2022. Retrieved 8 October 2022. ^ Bennett, Richard (September 2009). "Designed for Change: End-to-End Arguments, Internet Innovation, and the Net Neutrality Debate" (PDF). Information Technology and Innovation Foundation. pp. 7, 11. Retrieved 11 September 2017. ^ Abbate, Janet (2000). Inventing the Internet. MIT Press. pp. 124–127. ISBN 978-0-262-51115-5. In fact, CYCLADES, unlike ARPANET, had been explicitly designed to facilitate internetworking; it could, for instance, handle varying formats and varying levels of service ^ Kim, Byung-Keun (2005). Internationalising the Internet the Co-evolution of Influence and Technology. Edward Elgar. pp. 51–55. ISBN 1845426754. In addition to the NPL Network and the ARPANET, CYCLADES, an academic and research experimental network, also played an important role in the development of computer networking technologies ^ "The internet's fifth man". The Economist. 30 November 2013. ISSN 0013-0613. Retrieved 22 April 2020. In the early 1970s Mr Pouzin created an innovative data network that linked locations in France, Italy and Britain. Its simplicity and efficiency pointed the way to a network that could connect not just dozens of machines, but millions of them. It captured the imagination of Dr Cerf and Dr Kahn, who included aspects of its design in the protocols that now power the internet. ^ Moschovitis 1999, p. 78-9 ^ Cerf, V.; Kahn, R. (1974). "A Protocol for Packet Network Intercommunication" (PDF). IEEE Transactions on Communications. 22 (5): 637–648. doi:10.1109/TCOM.1974.1092259. ISSN 1558-0857. Archived (PDF) from the original on 6 January 2017. Retrieved 23 February 2020. The authors wish to thank a number of colleagues for helpful comments during early discussions of international network protocols, especially R. Metcalfe, R. Scantlebury, D. Walden, and H. Zimmerman; D. Davies and L. Pouzin who constructively commented on the fragmentation and accounting issues; and S. Crocker who commented on the creation and destruction of associations. ^ McKenzie, Alexander (2011). "INWG and the Conception of the Internet: An Eyewitness Account". IEEE Annals of the History of Computing. 33 (1): 66–71. doi:10.1109/MAHC.2011.9. ISSN 1934-1547. S2CID 206443072. ^ Schwartz, Mischa (2010). "X.25 Virtual Circuits - TRANSPAC IN France - Pre-Internet Data Networking [History of communications]". IEEE Communications Magazine. 48 (11): 40–46. doi:10.1109/MCOM.2010.5621965. ISSN 1558-1896. S2CID 23639680. ^ Rybczynski, Tony (2009). "Commercialization of packet switching (1975-1985): A Canadian perspective [History of Communications]". IEEE Communications Magazine. 47 (12): 26–31. doi:10.1109/MCOM.2009.5350364. ISSN 1558-1896. S2CID 23243636. ^ The "Hidden" Prehistory of European Research Networking. Trafford Publishing. p. 354. ISBN 978-1-4669-3935-6. ^ "TCP/IP Internet Protocol". Living Internet. Archived from the original on 1 September 2022. Retrieved 8 October 2022. ^ Andrew L. Russell (30 July 2013). "OSI: The Internet That Wasn't". IEEE Spectrum. Vol. 50, no. 8. ^ Russell, Andrew L. "Rough Consensus and Running Code' and the Internet-OSI Standards War" (PDF). IEEE Annals of the History of Computing. Archived (PDF) from the original on 17 November 2019. Retrieved 23 February 2020. ^ "Standards Wars" (PDF). 2006. Archived (PDF) from the original on 24 February 2021. Retrieved 23 February 2020. ^ Ben-Ari 1982, chapter 2 - The concurrent programming abstraction, p. 18-19, states the same. ^ Ben-Ari 1982, Section 2.7 - Summary, p. 27, summarizes the concurrent programming abstraction. ^ a b Marsden 1986, Section 6.1 - Why are standards necessary?, p. 64-65, uses BSC as an example to show the need for both standard protocols and a standard framework. ^ Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, explains this by drawing analogies between computer communication and programming languages. ^ a b Sect. 11.10 - The Disadvantage Of Layering, p. 192, states: layering forms the basis for protocol design. ^ a b Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, states the same. ^ Comer 2000, Sect. 11.3 - The Conceptual Layers Of Protocol Software, p. 178, "Each layer takes responsibility for handling one part of the problem." ^ Comer 2000, Sect. 11.11 - The Basic Idea Behind Multiplexing And Demultiplexing, p. 192, states the same. ^ "Data Communication - an overview". ScienceDirect Topics. Archived from the original on 31 May 2022. Retrieved 31 May 2022. ^ Kirch, Olaf (16 January 2002). "Text Based Protocols". Archived from the original on 30 May 2010. Retrieved 21 October 2014. ^ Kirch, Olaf (16 January 2002). "Binary Representation Protocols". Archived from the original on 30 May 2010. Retrieved 4 May 2006. ^ Kirch, Olaf (16 January 2002). "Binary Representation Protocols". Archived from the original on 5 March 2006. Retrieved 4 May 2006. ^ "Welcome To UML Web Site!". Uml.org. Archived from the original on 30 September 2019. Retrieved 15 January 2017. ^ Marsden 1986, Chapter 3 - Fundamental protocol concepts and problem areas, p. 26-42, explains much of the following. ^ Comer 2000, Sect. 7.7.4 - Datagram Size, Network MTU, and Fragmentation, p. 104, Explains fragmentation and the effect on the header of the fragments. ^ Comer 2000, Chapter 4 - Classful Internet Addresses, p. 64-67;71. ^ Marsden 1986, Section 14.3 - Layering concepts and general definitions, p. 187, explains address mapping. ^ Marsden 1986, Section 3.2 - Detection and transmission errors, p. 27, explains the advantages of backward error correction. ^ Marsden 1986, Section 3.3 - Acknowledgement, p. 28-33, explains the advantages of positive only acknowledgment and mentions datagram protocols as exceptions. ^ Marsden 1986, Section 3.4 - Loss of information - timeouts and retries, p. 33-34. ^ Marsden 1986, Section 3.5 - Direction of information flow, p. 34-35, explains master/slave and the negotiations to gain control. ^ Marsden 1986, Section 3.6 - Sequence control, p. 35-36, explains how packets get lost and how sequencing solves this. ^ Marsden 1986, Section 3.7 - Flow control, p. 36-38. ^ Ben-Ari 1982, in his preface, p. xiii. ^ Ben-Ari 1982, in his preface, p. xiv. ^ Hoare 1985, Chapter 4 - Communication, p. 133, deals with communication. ^ S. Srinivasan, Digital Circuits and Systems, NPTEL courses, archived from the original on 27 December 2009 ^ a b Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, introduces the decomposition in layers. ^ Comer 2000, Sect. 11.3 - The Conceptual Layers Of Protocol Software, p. 179, the first two paragraphs describe the sending of a message through successive layers. ^ Comer 2000, Sect. 11.2 - The need for multiple protocols, p. 178, explains similarities protocol software and compiler, assembler, linker, loader. ^ Comer 2000, Sect. 11.9.1 - Operating System Boundary, p. 192, describes the operating system boundary. ^ IETF 1989, Sect 1.3.1 - Organization, p. 15, 2nd paragraph: many design choices involve creative "breaking" of strict layering. ^ Comer 2000, Sect. 11.10 - The Disadvantage Of Layering, p. 192, explains why "strict layering can be extremely inefficient" giving examples of optimizations. ^ Wakeman, I (January 1992). "Layering considered harmful". IEEE Network: 20–24. ^ Kurose, James; Ross, Keith (2005). Computer Networking: A Top-Down Approach. Pearson. ^ Lascano, Jorge Edison; Clyde, Stephen; Raza, Ali. "Communication-protocol Design Patterns (CommDP) - COMMDP". Archived from the original on 18 March 2017. Retrieved 17 March 2017. ^ Lascano, J. E.; Clyde, S. (2016). A Pattern Language for Application-level Communication Protocols. ICSEA 2016, The Eleventh International Conference on Software Engineering Advances. pp. 22–30. ^ Daigneau, R. (2011). Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services (1 ed.). Upper Saddle River, NJ: Addison-Wesley Professional. ^ Fowler, M. (2002). Patterns of Enterprise Application Architecture (1 ed.). Boston: Addison-Wesley Professional. ISBN 0-321-12742-0. ^ [1]F. Buschmann, K. Henney, and D. C. Schmidt, Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing, Volume 4 edition. Chichester England; New York: Wiley, 2007. ^ Bochmann, G. (1978). "Finite state description of communication protocols". Computer Networks. 2 (4–5): 361–372. doi:10.1016/0376-5075(78)90015-6. ^ Comer 2000, Glossary of Internetworking Terms and Abbreviations, p. 704, term protocol. ^ Brand, Daniel; Zafiropulo, Pitro (1983). "On Communicating Finite-State Machines". Journal of the ACM. 30 (2): 323. doi:10.1145/322374.322380. S2CID 11607967. ^ Marsden 1986, Section 6.3 - Advantages of standardization, p. 66-67, states the same. ^ Bryant & Morrow 2009, p. 4. ^ Marsden 1986, Section 6.4 - Some problems with standardisation, p. 67, follows HDLC to illustrate the process. ^ "X.225 : Information technology – Open Systems Interconnection – Connection-oriented Session protocol: Protocol specification". Archived from the original on 1 February 2021. Retrieved 10 March 2023. ^ Marsden 1986, Section 6.1 - Why are standards necessary?, p. 65, explains lessons learned from ARPANET. ^ Marsden 1986, Section 14.1 - Introduction, p. 181, introduces OSI. ^ Marsden 1986, Section 14.3 – Layering concepts and general definitions, p. 183-185, explains terminology. ^ Marsden 1986, Section 14.4 - The application layer, p. 188, explains this. ^ Marsden 1986, Section 14.5 - The presentation layer, p. 189, explains this. ^ Marsden 1986, Section 14.6 - The session layer, p. 190, explains this. ^ Marsden 1986, Section 14.7 - The transport layer, p. 191, explains this. ^ Marsden 1986, Section 14.8 - The network layer, p. 192, explains this. ^ Marsden 1986, Section 14.9 - The data link layer, p. 194, explains this. ^ Marsden 1986, Section 14.10 - The physical layer, p. 195, explains this. ^ ISO 7498:1984 – Information processing systems - Open Systems Interconnection - Basic Reference Model. p. 5. This Basic Reference Model of Open Systems Interconnection is based on the assumption that a connection is required for the transfer of data. ^ ISO 7498:1984/ADD 1:1987 – Information processing systems — Open Systems Interconnection — Basic Reference Model — Addendum 1. ^ Marsden 1986, Section 14.11 - Connectionless mode and RM/OSI, p. 195, mentions this. ^ ISO 7498:1994 – Information processing systems - Open Systems Interconnection - Basic Reference Model. ^ Comer 2000, Section 1.9 - Internet Protocols And Standardization, p. 12, explains why the IETF did not use existing protocols. ^ a b Trammell & Kuehlewind 2019, p. 2. ^ a b Trammell & Kuehlewind 2019, p. 3. ^ Trammell & Kuehlewind 2019, p. 4. ^ a b Fairhurst & Perkins 2021, 7. Conclusions. ^ Trammell & Kuehlewind 2019, p. 5. ^ Trammell & Kuehlewind 2019, p. 6. ^ Trammell & Kuehlewind 2019, p. 7-8. ^ Farrell & Tschofenig 2014, p. 2. ^ a b Farrell & Tschofenig 2014, p. 3. ^ Arkko et al. 2023, 2.1. Intentional Distribution. ^ Arkko et al. 2023, 2.2. Control of the Distribution of Information. ^ Arkko et al. 2023, 2.3. Protecting Information and Authentication. ^ Arkko et al. 2023, 2.5. Limiting Impact of Information. ^ Arkko et al. 2023, 2.4. Minimize Information. ^ Arkko et al. 2023, 2.6. Minimum Set of Entities. ^ Arkko et al. 2023, 3. Further Work. ^ Papastergiou et al. 2017, p. 619. ^ Papastergiou et al. 2017, p. 620. ^ Papastergiou et al. 2017, p. 620-621. ^ Papastergiou et al. 2017, p. 623-4. ^ McQuistin, Perkins & Fayed 2016, p. 1. ^ Thomson & Pauly 2021, A.5. TCP. ^ Hardie 2019, p. 7-8. ^ Thomson & Pauly 2021, 3. Active Use. ^ Thomson & Pauly 2021, 3.5. Restoring Active Use. ^ Comer 2000, Sect. 11.5.1 - The TCP/IP 5-Layer Reference Model, p. 183, states the same. Radia Perlman (1999). Interconnections: Bridges, Routers, Switches, and Internetworking Protocols (2nd ed.). Addison-Wesley. ISBN 0-201-63448-1.. In particular Ch. 18 on "network design folklore", which is also available online Gerard J. Holzmann (1991). Design and Validation of Computer Protocols. Prentice Hall. ISBN 0-13-539925-4. Douglas E. Comer (2000). Internetworking with TCP/IP - Principles, Protocols and Architecture (4th ed.). Prentice Hall. ISBN 0-13-018380-6. In particular Ch.11 Protocol layering. Also has a RFC guide and a Glossary of Internetworking Terms and Abbreviations. R. Braden, ed. (1989). Requirements for Internet Hosts -- Communication Layers. Internet Engineering Task Force abbr. IETF. doi:10.17487/RFC1122. RFC 1122. Describes TCP/IP to the implementors of protocolsoftware. In particular the introduction gives an overview of the design goals of the suite. M. Ben-ari (1982). Principles of concurrent programming (10th Print ed.). Prentice Hall International. ISBN 0-13-701078-8. C.A.R. Hoare (1985). Communicating sequential processes (10th Print ed.). Prentice Hall International. ISBN 0-13-153271-5. R.D. Tennent (1981). Principles of programming languages (10th Print ed.). Prentice Hall International. ISBN 0-13-709873-1. Brian W Marsden (1986). Communication network protocols (2nd ed.). Chartwell Bratt. ISBN 0-86238-106-1. Andrew S. Tanenbaum (1984). Structured computer organization (10th Print ed.). Prentice Hall International. ISBN 0-13-854605-3. Bryant, Stewart; Morrow, Monique, eds. (November 2009). Uncoordinated Protocol Development Considered Harmful. doi:10.17487/RFC5704. RFC 5704. Farrell, Stephen; Tschofenig, Hannes (May 2014). Pervasive Monitoring Is an Attack. doi:10.17487/RFC7258. RFC 7258. Trammell, Brian; Kuehlewind, Mirja (April 2019). The Wire Image of a Network Protocol. doi:10.17487/RFC8546. RFC 8546. Hardie, Ted, ed. (April 2019). Transport Protocol Path Signals. doi:10.17487/RFC8558. RFC 8558. Fairhurst, Gorry; Perkins, Colin (July 2021). Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols. doi:10.17487/RFC9065. RFC 9065. Thomson, Martin; Pauly, Tommy (December 2021). Long-Term Viability of Protocol Extension Mechanisms. doi:10.17487/RFC9170. RFC 9170. Arkko, Jari; Hardie, Ted; Pauly, Tommy; Kühlewind, Mirja (July 2023). Considerations on Application - Network Collaboration Using Path Signals. doi:10.17487/RFC9419. RFC 9419. McQuistin, Stephen; Perkins, Colin; Fayed, Marwan (July 2016). Implementing Real-Time Transport Services over an Ossified Network. 2016 Applied Networking Research Workshop. doi:10.1145/2959424.2959443. hdl:1893/26111. Papastergiou, Giorgos; Fairhurst, Gorry; Ros, David; Brunstrom, Anna; Grinnemo, Karl-Johan; Hurtig, Per; Khademi, Naeem; Tüxen, Michael; Welzl, Michael; Damjanovic, Dragana; Mangiante, Simone (2017). "De-Ossifying the Internet Transport Layer: A Survey and Future Perspectives". IEEE Communications Surveys & Tutorials. 19: 619–639. doi:10.1109/COMST.2016.2626780. hdl:2164/8317. S2CID 1846371. Moschovitis, Christos J. P. (1999). History of the Internet: A Chronology, 1843 to the Present. ABC-CLIO. ISBN 978-1-57607-118-2. Javvin's Protocol Dictionary at the Wayback Machine (archived 2004-06-10) Overview of protocols in telecontrol field with OSI Reference Model Retrieved from "