Communication protocol

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.[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 pre-determined 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.

Communicating systems

History

One of the first uses of the term *protocol* in a data-commutation context occurs in a memorandum entitled *A Protocol for Use in the NPL Data Communications Network* written by Roger Scantlebury and Keith Bartlett in April 1967.[5][6]

On the ARPANET, the starting point for host-to-host communication in 1969 was the 1822 protocol, which defined the transmission of messages to an IMP.[7] The Network Control Protocol (NCP) for

the ARPANET was first implemented in 1970.[8] 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.[9]

Networking research in the early 1970s by Robert E. Kahn and Vint Cerf led to the formulation of the Transmission Control Program (TCP).[10] Its RFC 675 (https://datatracker.ietf.org/doc/html/rfc675) specification was written by Cerf with Yogen Dalal and Carl Sunshine in December 1974, still a monolithic design at this time.

The International Networking Working Group agreed a connectionless datagram standard which was presented to the CCIT in 1975 but was not adopted by the ITU or by the ARPANET.[11] International research, particularly the work of Rémi Després, contributed to the development of the X.25 standard, based on virtual circuits by the ITU-T in 1976.[12][13] Computer manufacturers developed proprietary protocols such as IBM's Systems Network Architecture (SNA), Digital Equipment Corporation's DECnet and Xerox Network Systems.[14]

TCP software was redesigned as a modular protocol stack. Originally referred to as *IP/TCP*, it was installed on SATNET in 1982 and on the ARPANET in January 1983. The development of a complete protocol suite by 1989, as outlined in RFC 1122 (https://datatracker.ietf.org/doc/html/rfc1122) and RFC 1123 (https://datatracker.ietf.org/doc/html/rfc1123) , laid the foundation for the growth of TCP/IP as a comprehensive protocol suite as the core component of the emerging Internet.[15]

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.[16][17][18]

Concept

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.[19][20] 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.[21] 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.[22] This gave rise to the concept of layered protocols which nowadays forms the basis of protocol design.[23]

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.[24] 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.[25] 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.[26]

Types

There are two types of communication protocols, based on their representation of the content being carried: text-based and binary.[27]

Text-based

A **text-based protocol** or **plain text protocol** represents its content in human-

readable format, often in plain text.

The immediate human readability stands in contrast to 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), and the finger protocol.[28]

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.

To be clear, all digital communication is fundamentally binary. The "Text" based

protocols mentioned here use only binary content, which is made "humanly readable" by a text editor (or other such software).

Binary

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.[29]

Binary have been used in the normative documents describing modern standards like EbXML, HTTP/2, HTTP/3 and EDOC.[30] An interface in UML[31] may also be considered a binary protocol.

Basic requirements

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:[32]

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.[33]

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-*1*s address could be taken to mean an addressing of all stations on the network, so sending to this address would result in a broadcast on the local network. The rules describing the meanings of the address value are collectively called an *addressing scheme*.[34]

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*.[35]

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.[36]

Acknowledgements

Acknowledgement of correct reception of packets is required for connection-oriented communication.

Acknowledgments are sent from receivers back to their respective senders.[37]

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.[38]

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.[39]

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.[40]

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.[41]

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.

Protocol design

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,[24] 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.[42] Formal verification seems indispensable because concurrent programs are notorious for the hidden and sophisticated bugs they contain.[43] A mathematical approach to the study of

concurrency and communication is referred to as communicating sequential processes (CSP).[44] 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.[45]

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.

Layering

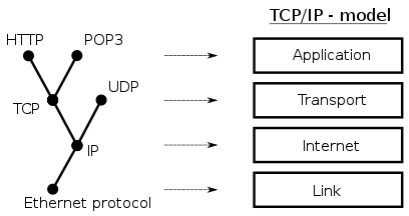

Figure 2. The TCP/IP model or Internet layering scheme and its relation to some common protocols.

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.[46] 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.

Protocol layering

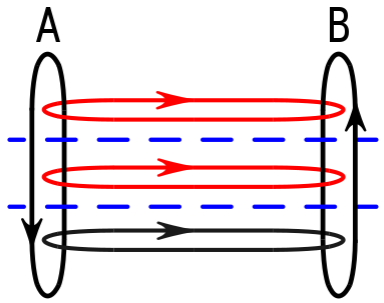

Figure 3. Message flows using a protocol suite. Black loops show the actual messaging loops, red loops are the effective communication between layers enabled by the lower layers.

Protocol layering forms the basis of protocol design.[23] It allows the decomposition of single, complex protocols into simpler, cooperating protocols.[46] 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.

Software layering

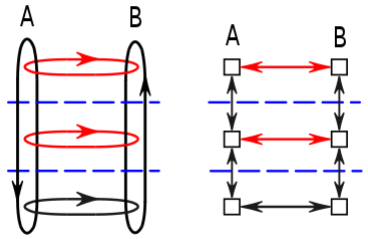

Figure 5: Protocol and software layering. The software modules implementing the protocols are represented by cubes. The information flow between the modules is represented by arrows. The (top two horizontal) red arrows are virtual. The blue lines mark the layer boundaries.

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.[47]

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.[48]

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.[49]

Strict layering

Strictly adhering to a layered model, a practice known as strict layering, is not always the best approach to networking.[50] Strict layering can have a negative impact on the performance of an implementation.[51]

While the use of protocol layering is today ubiquitous across the field of computer networking, it has been historically criticized by many researchers[52] 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.[53]

Design patterns

Commonly recurring problems in the design and implementation of communication protocols can be addressed by software design patterns.[54][55][56][57][58]

Formal specification

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.[59][60] and communicating finite-state machines[61]

Protocol development

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

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.[21]

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.

Standards organizations

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.[62]

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.[63]

The standardization process

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.[64]

OSI standardization

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).[66] 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.[67]

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.[68] 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.[69]

- 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).[70]

- 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.[71]

- 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.[72]

- 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.[73]
- 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.[74]

- 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.[75]

In contrast to the TCP/IP layering scheme, which assumes a connectionless network, RM/OSI assumed a connection-oriented network.[76] 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[77][78] and later incorporated into an update to RM/OSI.[79]

At the time, the IETF had to cope with this and the fact that the Internet needed protocols that simply were not there. As a result, the IETF developed its own standardization process based on "rough consensus and running code".[80] The standardization process is described by RFC 2026 (https://datatracker.ietf.org/doc/html/rfc2026) .

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.

Wire image

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.[81] Unencrypted protocol metadata is one source making up the wire image, and side-channels including packet timing also contribute.[82] Different observers with different vantages may see different wire images.[83] The wire image is

relevant to end-user privacy and the extensibility of the protocol.[84]

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.[82] 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.[85]

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.[86] If provided signals are decoupled from the protocol's operation, they may become untrustworthy.[87] 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.[84] 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,[88] and that the IETF would "work to mitigate pervasive monitoring" in its protocol designs;[89] this had not been done systematically previously.[89]

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.[90] This is a violation

of the end-to-end principle.[91] Secondary causes include inflexibility in endpoint implementations of protocols.[92]

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.[93] Because of ossification, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the only practical

choices for transport protocols on the Internet,[94] and TCP itself has significantly ossified, making extension or modification of the protocol difficult.[95]

Recommended methods of preventing ossification include encrypting protocol metadata,[96] and ensuring that extension points are exercised and wire image variability is exhibited as fully as possible;[97] remedying existing ossification requires coordination across protocol participants.[98] QUIC is the first IETF transport protocol to have been designed with deliberate anti-ossification properties.[81]

Taxonomies

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.[99] 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.

See also

- Lists of network protocols
- Protocol Builder

Notes

a. *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.*

References

1. *US 7529565 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US7529565) , Hilpisch, Robert E.; Duchscher, Rob & Seel, Mark et al., "Wireless communication*

protocol", published 2009-05-05, assigned to Starkey Laboratories Inc. and Oticon AS

2. *Protocol (https://www.britannica.com/EBchecked/topic/410357/protocol) , Encyclopædia Britannica, retrieved 24 September 2012*
3. *Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, "They (protocols) are to communication what programming languages are to computation"*
4. *Comer 2000, Sect. 1.3 - Internet Services, p. 3, "Protocols are to communication what algorithms are to computation"*
5. *Naughton, John (24 September 2015). A Brief History of the Future (https://books.google.com/books?id=bbonCgAAQBAJ&pg=PT290) . Orion. ISBN 978-1-4746-0277-8.*

6. *Cambell-Kelly, Martin (1987). "Data Communications at the National Physical Laboratory (1965-1975)" (https://archive.org/details/DataCommunicationsAtTheNationalPhysicalLaboratory) . Annals of the History of Computing. **9** (3/4): 221–247. doi:10.1109/MAHC.1987.10023 (https://doi.org/10.1109%2FMAHC.1987.10023) . S2CID 8172150 (https://api.semanticscholar.org/CorpusID:8172150) .*
7. *Interface Message Processor: Specifications for the Interconnection of a Host and an IMP (http://www.bitsavers.org/pdf/bbn/imp/BBN1822_Jan1976.pdf) (PDF) (Report). Bolt Beranek and Newman (BBN). Report No. 1822.*

8. *BOOKS, HIGH DEFINITION. UGC - NET/JRF/SET PTP & Guide Teaching and Research Aptitude: UGC -NET By HD (https://books.google.com/books?id=dRRDDwAAQBAJ&pg=PA319)* . *High Definition Books.*
9. *"NCP – Network Control Program" (https://livinginternet.com/i/ii_ncp.htm)* . *Living Internet.*

10. *Cerf, V.; Kahn, R. (1974). "A Protocol for Packet Network Intercommunication" (https://www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf) (PDF). IEEE Transactions on Communications.* ***22*** *(5): 637–648. doi:10.1109/TCOM.1974.1092259 (https://doi.org/10.1109%2FTCOM.1974.1092259) . ISSN 1558-0857 (https://www.worldcat.org/issn/1558-0857) . "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."

11. *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 (https://doi.org/10.1109%2FMAHC.2011.9) . ISSN 1934-1547 (https://www.worldcat.org/issn/1934-1547) . S2CID 206443072 (https://api.semanticscholar.org/CorpusID:206443072) .*

12. *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 (https://doi.org/10.1109%2FMCOM.2010.5621965) . ISSN 1558-1896 (https://www.worldcat.org/issn/1558-1896) . S2CID 23639680 (https://api.semanticscholar.org/CorpusID:23639680) .*

13. *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 (https://doi.org/10.1109%2FMCOM.2009.5350364) . ISSN 1558-1896 (https://www.worldcat.org/issn/1558-1896) . S2CID 23243636 (https://api.semanticscholar.org/CorpusID:23243636) .*
14. *The "Hidden" Prehistory of European Research Networking (https://books.google.com/books?id=y-tFrTTweRIC&pg=PA337) . Trafford Publishing. p. 354. ISBN 978-1-4669-3935-6.*

15. *"TCP/IP Internet Protocol" (https://livinginternet.com/i/ii_tcpip.htm)* . *Living Internet*.
16. *Andrew L. Russell (30 July 2013). "OSI: The Internet That Wasn't" (https://spectrum.ieee.org/computing/networks/osi-the-internet-that-wasnt)* . *IEEE Spectrum*. *Vol. 50, no. 8*.
17. *Russell, Andrew L. "Rough Consensus and Running Code' and the Internet-OSI Standards War" (https://www2.cs.duke.edu/courses/common/compsci092/papers/govern/consensus.pdf)* *(PDF)*. *IEEE Annals of the History of Computing*.
18. *"Standards Wars" (https://courses.cs.washington.edu/courses/csep590a/06au/projects/standards-wars.pdf)* *(PDF)*. *2006*.

19. *Ben-Ari 1982, chapter 2 - The concurrent programming abstraction, p. 18-19, states the same.*
20. *Ben-Ari 1982, Section 2.7 - Summary, p. 27, summarizes the concurrent programming abstraction.*
21. *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.*
22. *Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, explains this by drawing analogies between computer communication and programming languages.*

23. *Sect. 11.10 - The Disadvantage Of Layering, p. 192, states: layering forms the basis for protocol design.*
24. *Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, states the same.*
25. *Comer 2000, Sect. 11.3 - The Conceptual Layers Of Protocol Software, p. 178, "Each layer takes responsibility for handling one part of the problem."*
26. *Comer 2000, Sect. 11.11 - The Basic Idea Behind Multiplexing And Demultiplexing, p. 192, states the same.*
27. *"Data Communication - an overview | ScienceDirect Topics" (https://www.sciencedirect.com/topics/computer-science/data-communication) . www.sciencedirect.com. Retrieved 31 May 2022.*

28. *Kirch, Olaf (16 January 2002). "Text Based Protocols" (https://web.archive.org/web/20100530140215/http://www.lst.de/~okir/blackhats/node76.html) . Archived from the original (http://www.lst.de/~okir/blackhats/node76.html) on 30 May 2010. Retrieved 21 October 2014.*
29. *Kirch, Olaf (16 January 2002). "Binary Representation Protocols" (https://web.archive.org/web/20100530225453/http://www.lst.de/~okir/blackhats/node77.html) . Archived from the original (http://www.lst.de/~okir/blackhats/node77.html) on 5 March 2006. Retrieved 4 May 2006.*

30. *Kirch, Olaf (16 January 2002). "Binary Representation Protocols" (https://web.archive.org/web/20060305095832/http://www.omg.org/technology/documents/formal/edoc.htm) . Archived from the original (http://www.omg.org/technology/documents/formal/edoc.htm) on 5 March 2006. Retrieved 4 May 2006.*
31. *"Welcome To UML Web Site!" (https://www.uml.org/) . Uml.org. Retrieved 15 January 2017.*
32. *Marsden 1986, Chapter 3 - Fundamental protocol concepts and problem areas, p. 26-42, explains much of the following.*

33. *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.*
34. *Comer 2000, Chapter 4 - Classful Internet Addresses, p. 64-67;71.*
35. *Marsden 1986, Section 14.3 - Layering concepts and general definitions, p. 187, explains address mapping.*
36. *Marsden 1986, Section 3.2 - Detection and transmission errors, p. 27, explains the advantages of backward error correction.*
37. *Marsden 1986, Section 3.3 - Acknowledgement, p. 28-33, explains the advantages of positive only acknowledgment and mentions datagram protocols as exceptions.*

38. *Marsden 1986, Section 3.4 - Loss of information - timeouts and retries, p. 33-34.*
39. *Marsden 1986, Section 3.5 - Direction of information flow, p. 34-35, explains master/slave and the negotiations to gain control.*
40. *Marsden 1986, Section 3.6 - Sequence control, p. 35-36, explains how packets get lost and how sequencing solves this.*
41. *Marsden 1986, Section 3.7 - Flow control, p. 36-38.*
42. *Ben-Ari 1982, in his preface, p. xiii.*
43. *Ben-Ari 1982, in his preface, p. xiv.*
44. *Hoare 1985, Chapter 4 - Communication, p. 133, deals with communication.*

45. *S. Srinivasan, Digital Circuits and Systems (https://web.archive.org/web/20091227210642/http://nptel.iitm.ac.in/video.php?courseId=1005&p=3) , NPTEL courses, archived from the original (http://nptel.iitm.ac.in/video.php?courseId=1005&p=3) on 27 December 2009*
46. *Comer 2000, Sect. 11.2 - The Need For Multiple Protocols, p. 177, introduces the decomposition in layers.*
47. *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.*

48. *Comer 2000, Sect. 11.2 - The need for multiple protocols, p. 178, explains similarities protocol software and compiler, assembler, linker, loader.*
49. *Comer 2000, Sect. 11.9.1 - Operating System Boundary, p. 192, describes the operating system boundary.*
50. *IETF 1989, Sect 1.3.1 - Organization, p. 15, 2nd paragraph: many design choices involve creative "breaking" of strict layering.*
51. *Comer 2000, Sect. 11.10 - The Disadvantage Of Layering, p. 192, explains why "strict layering can be extremely inefficient" giving examples of optimizations.*
52. *Wakeman, I (January 1992). "Layering considered harmful". IEEE Network: 20–24.*

53. *Kurose, James; Ross, Keith (2005). Computer Networking: A Top-Down Approach. Pearson.*
54. *Lascano, Jorge Edison; Clyde, Stephen; Raza, Ali. "Communication-protocol Design Patterns (CommDP) - COMMDP" (https://web.archive.org/web/20170318090654/http://commdp.serv.usu.edu/wiki/index.php/Communication-protocol_Design_Patterns_(CommDP)) . Archived from the original (http://commdp.serv.usu.edu/wiki/index.php/Communication-protocol_Design_Patterns_(CommDP)) on 18 March 2017. Retrieved 17 March 2017.*

55. *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.*
56. *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.*
57. *Fowler, M. (2002). Patterns of Enterprise Application Architecture (1 ed.). Boston: Addison-Wesley Professional. ISBN 0-321-12742-0.*

58. *[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.*
59. *Bochmann, G. (1978). "Finite state description of communication protocols". Computer Networks.* ***2*** *(4–5): 361–372. doi:10.1016/0376-5075(78)90015-6 (https://doi.org/10.1016%2F0376-5075%2878%2990015-6) .*
60. *Comer 2000, Glossary of Internetworking Terms and Abbreviations, p. 704, term protocol.*

61. *Brand, Daniel; Zafiropulo, Pitro (1983). "On Communicating Finite-State Machines". Journal of the ACM. **30** (2): 323. doi:10.1145/322374.322380 (https://doi.org/10.1145%2F322374.322380) . S2CID 11607967 (https://api.semanticscholar.org/CorpusID:11607967) .*
62. *Marsden 1986, Section 6.3 - Advantages of standardization, p. 66-67, states the same.*
63. *Bryant & Morrow 2009, p. 4.*
64. *Marsden 1986, Section 6.4 - Some problems with standardisation, p. 67, follows HDLC to illustrate the process.*

65. *"X.225 : Information technology – Open Systems Interconnection – Connection-oriented Session protocol: Protocol specification" (https://www.itu.int/rec/T-REC-X.225-199511-I/en) . Archived (https://web.archive.org/web/20210201064044/https://www.itu.int/rec/T-REC-X.225-199511-I/en) from the original on 1 February 2021. Retrieved 10 March 2023.*
66. *Marsden 1986, Section 6.1 - Why are standards necessary?, p. 65, explains lessons learned from ARPANET.*
67. *Marsden 1986, Section 14.1 - Introduction, p. 181, introduces OSI.*
68. *Marsden 1986, Section 14.3 - Layering concepts and general definitions, p. 183-185, explains terminology.*

69. *Marsden 1986, Section 14.4 - The application layer, p. 188, explains this.*
70. *Marsden 1986, Section 14.5 - The presentation layer, p. 189, explains this.*
71. *Marsden 1986, Section 14.6 - The session layer, p. 190, explains this.*
72. *Marsden 1986, Section 14.7 - The transport layer, p. 191, explains this.*
73. *Marsden 1986, Section 14.8 - The network layer, p. 192, explains this.*
74. *Marsden 1986, Section 14.9 - The data link layer, p. 194, explains this.*
75. *Marsden 1986, Section 14.10 - The physical layer, p. 195, explains this.*

76. *ISO 7498:1984 – Information processing systems - Open Systems Interconnection - Basic Reference Model (https://www.iso.org/standard/14252.html) . 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."*
77. *ISO 7498:1984/ADD 1:1987 – Information processing systems — Open Systems Interconnection — Basic Reference Model — Addendum 1 (https://www.iso.org/standard/14253.html) .*
78. *Marsden 1986, Section 14.11 - Connectionless mode and RM/OSI, p. 195, mentions this.*

79. *ISO 7498:1994 – Information processing systems - Open Systems Interconnection - Basic Reference Model (https://www.iso.org/standard/20269.html)* .
80. *Comer 2000, Section 1.9 - Internet Protocols And Standardization, p. 12, explains why the IETF did not use existing protocols.*
81. *Trammell & Kuehlewind 2019, p. 2.*
82. *Trammell & Kuehlewind 2019, p. 3.*
83. *Trammell & Kuehlewind 2019, p. 4.*
84. *Fairhurst & Perkins 2021, 7. Conclusions.*
85. *Trammell & Kuehlewind 2019, p. 5.*
86. *Trammell & Kuehlewind 2019, p. 6.*
87. *Trammell & Kuehlewind 2019, p. 7-8.*
88. *Farrell & Tschofenig 2014, p. 2.*

89. *Farrell & Tschofenig 2014, p. 3.*
90. *Papastergiou et al. 2017, p. 619.*
91. *Papastergiou et al. 2017, p. 620.*
92. *Papastergiou et al. 2017, p. 620-621.*
93. *Papastergiou et al. 2017, p. 623-4.*
94. *McQuistin, Perkins & Fayed 2016, p. 1.*
95. *Thomson & Pauly 2021, A.5. TCP.*
96. *Hardie 2019, p. 7-8.*
97. *Thomson & Pauly 2021, 3. Active Use.*
98. *Thomson & Pauly 2021, 3.5. Restoring Active Use.*
99. *Comer 2000, Sect. 11.5.1 - The TCP/IP 5-Layer Reference Model, p. 183, states the same.*

Bibliography

- 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 (https://www.informit.com/articles/article.aspx?p=20482)
- Gerard J. Holzmann (1991). *Design and Validation of Computer Protocols* (http://spinroot.com/spin/Doc/Book91.html) . 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* (https://datatracker.ietf.org/doc/html/rfc1122) . Internet Engineering Task Force abbr. IETF. doi:10.17487/RFC1122 (https://doi.org/10.17487%2FRFC1122) . RFC 1122 (https://datatracker.ietf.org/d

oc/html/rfc1122) . Describes TCP/IP to the implementors of protocolsoftware. In particular the introduction gives an overview of the design goals of the suite.

- M. Ben-ari (1982). *Principles of concurrent programming* (10th Print ed.). Prentice Hall International. ISBN 0-13-701078-8.
- C.A.R. Hoare (1985). *Communicating sequential processes* (http://www.usingcsp.com) (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* (https://datatracker.ietf.org/doc/html/rfc5704) .

doi:10.17487/RFC5704 (https://doi.org/10.17487%2FRFC5704) . RFC 5704 (https://datatracker.ietf.org/doc/html/rfc5704) .

- Farrell, Stephen; Tschofenig, Hannes (May 2014). *Pervasive Monitoring Is an Attack* (https://datatracker.ietf.org/doc/html/rfc7258) . doi:10.17487/RFC7258 (https://doi.org/10.17487%2FRFC7258) . RFC 7258 (https://datatracker.ietf.org/doc/html/rfc7258) .
- Trammell, Brian; Kuehlewind, Mirja (April 2019). *The Wire Image of a Network Protocol* (https://datatracker.ietf.org/doc/html/rfc8546) .

doi:10.17487/RFC8546 (https://doi.org/10.17487%2FRFC8546) . RFC 8546 (https://datatracker.ietf.org/doc/html/rfc8546) .

- Hardie, Ted, ed. (April 2019). *Transport Protocol Path Signals* (https://datatracker.ietf.org/doc/html/rfc8558) . doi:10.17487/RFC8558 (https://doi.org/10.17487%2FRFC8558) . RFC 8558 (https://datatracker.ietf.org/doc/html/rfc8558) .
- Fairhurst, Gorry; Perkins, Colin (July 2021). *Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet*

Transport Protocols (https://datatracker.ietf.org/doc/html/rfc9065) . doi:10.17487/RFC9065 (https://doi.org/10.17487%2FRFC9065) . RFC 9065 (https://datatracker.ietf.org/doc/html/rfc9065) .

- Thomson, Martin; Pauly, Tommy (December 2021). *Long-Term Viability of Protocol Extension Mechanisms* (https://datatracker.ietf.org/doc/html/rfc9170) . doi:10.17487/RFC9170 (https://doi.org/10.17487%2FRFC9170) . RFC 9170 (https://datatracker.ietf.org/doc/html/rfc9170) .

- 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 (https://doi.org/10.1145%2F2959424.2959443) . hdl:1893/26111 (https://hdl.handle.net/1893%2F26111) .
- 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 (https://doi.org/10.1109%2FCOMST.2016.2626780) . hdl:2164/8317 (https://hdl.handle.net/2164%2F8317) . S2CID 1846371 (https://api.semanticscholar.org/CorpusID:1846371) .

External links

- Javvin's Protocol Dictionary (https://web.archive.org/web/20040610001039/http://javvin.com/protocolsuite.html)

- Overview of protocols in telecontrol field with OSI Reference Model (https://www.ipcomm.de/protocols_en.html)