Time, reliable communications, distributed algorithms and the Internet

Gérard Le Lann

Time, reliable communications, distributed algorithms and the Internet

Gérard Le Lann, Research Director Emeritus, INRIA Paris-Rocquencourt, France

October 21, 2021

Translated from the original article in French (in Revue Telecoms, Issue 201, Aug. 2021) for the InterPlanetary Networking Special Interest Group (IPNSIG, https://ipnsig.org/), at the invitation of Vint Cerf.

I am grateful to Vint Cerf and Bob Kahn for their proofreading of this article.

0. Introduction

Is the use of physical time inevitable in any protocol intended to guarantee reliable communications in lossy networks ("best effort")? What are the links between TCP/IP and distributed algorithms at the heart of contemporary networks?

The first question is still the subject of debate among specialists. The second question is very rarely asked, despite its historical scientific importance.

1. Transport protocols for “best effort” type networks

We consider unreliable (“best effort”) packet switched networks [1], with variable delays. We assume a Newtonian spacetime referential. The packets flow in any order, and multiple copies of the same packet can exist. Let *R* be such a network.

Let us consider a source *S* and a recipient *D*, sequential processes executed on devices connected to *R*, and a virtual link *LV* (transport level, OSI/ISO 4) on which *S* sends an ordered sequence (Ω) of messages to *D*. By reliable communications, we mean delivery of Ω to *D*.

TCP [2] and its sliding window scheme, strongly influenced by the simulation work conducted at University of Rennes in 1972 and 1973—member of the Cyclades project (see reference [8] in [2]), along with work that led to IP, are the basis for TCP/IP as seen through today's eyes [3]. According to traditional presentations, a transport protocol aims to ensure the following properties:

- Delivery to D of a series of messages strictly identical to Ω (ordered, without losses, without repetitions),

- By optimizing the use of the resources available at S, R and D, without saturating them.

From an algorithmic point of view, TCP’s aim is more ambitious.

Let *X* be a message received and acknowledged by *D*. We want to ensure the following **agreement property Ψ**:

- Proposition Φ(*S*): *S* knows that *D* has received *X*.

- Proposition Φ(*D*): *D* knows that *S* knows that *D* has received *X*.

There is agreement between *S* and *D* when Φ(*S*) and Φ(*D*) are both true.

The TCP protocol and its variants — including the most recent ones [4] — guarantee Ψ, thanks to the “end-to-end” error and flow control exerted by the sliding window mechanism, based on acks or nacks, and timers, whose values are inferred from the *RTT* variable. *RTT* is a function of Δ_S, Δ_R and Δ_D. Δ_S and Δ_D are the activation and execution durations of *S* and *D* on their respective devices. Δ_R is the transit delay of *R* [5].

A message is segmented into Protocol Data Units (PDUs) of maximum known length (necessary for flow control), transmitted by *R* in the form of a packet (IPv4 or IPv6).

S sends numbered PDUs to *D* (consecutive integers *j*, modulo a base ignored here) and *D* sends ack (*j*) or nack (*j*) to *S*. *S*, *D*, and *R* are subject to transient or permanent faults (shutdowns for *S* and *D*, loss of ack/nack or PDU — an erroneous PDU is ignored).

It is assumed that *S* and *D* are honest (no impersonation, no tampering with PDUs, etc.). The resources allocated to *LV* (processing time, memory, input-output channels, etc.) are shared with other processes present in the devices and in *R*.

At any time, we have (time index ignored):

- *w*: size of the current window on *LV*, known identically by *S* and *D*; *w* is dynamically adjustable according to the observed loads and delays [6].

- $j(S)$: all PDUs with numbers up to $j(S)$ have been acknowledged by *D* (according to *S*) / the lower edge of the window at *S* is $j^* = j(S) + 1$

==> Besides number $j \in [j(S) + 1, j(S) + 1 + w]$, any PDU is accompanied by j^*.

- $j(D)$: all PDUs with numbers up to $j(D)$ have been acknowledged by *D* (according to *D*) / the lower edge of the window at *D* is $j(D) + 1$

==> *D* accepts PDUs numbered from $j(D) + 1$ to $j(D) + 1 + w$.

Unless lost, ack/nack are received by *S* after a certain delay. So, $j(S) \leq j(D)$. In the worst case, *S* refrains from sending a total of $j(D)$-$j(S)$ PDUs while *D* is ready to receive them (loss of efficiency).

This simplified description encompasses all variants of ARQ protocols [7], those which are based on acks and retransmissions of all PDUs whose numbers belong to the current window, as well as those which use nacks (triggering retransmissions of PDU number $j(S) + 1$ or PDUs which have been negatively acknowledged by *D*). A maximum number of retransmissions (PDU, ack, nack) is set. When this threshold is reached, *LV* is closed.

As long as *LV* is not closed, *D* must return within a known maximum delay (a function of *RTT*) an ack for the PDUs received, or a nack in the event of a missing PDU, possibly repeated. After receiving an ack/nack, *S* must send one or more PDUs (possibly an empty PDU) within a known maximum delay (a function of *RTT*).

Back to the **agreement property Ψ** (last PDU for message *X* is PDU *M* numbered *m*, received and acknowledged by *D*):

- The proposition $\Phi(S)$ is true when *S* receives ack(v), $v \geq m$, or nack(v), $v \geq m + 1$.

- The proposition $\Phi(D)$ is true when *D* receives a PDU carrying a number $j^* \geq m$.

2. Physical time and transport protocols

We examine the following **assertion A**:

The use of physical time is inevitable in any protocol intended to ensure Ψ.

Some experts (excellent connoisseurs of the history of network protocols) try to demonstrate that A is incorrect via a counterexample built with the ABP protocol [8], which is based on flooding the resources. A single message (bit $b = 0$ for example) is in transit between *S* and *D*. *S* "floods" *R* and *D*, continuously sending copies of the message in transit, until an ack (b) is received. The following message is then sent by *S*, stamped with bit $b = 1$. And so on. No timer is used in ABP.

2.1. Discussion from the perspective of "protocol engineering"

a) ABP is an OSI/ISO level 2 (data link) protocol

S and *D* communicate directly via a physical link (no network such as *R*), as considered in the NPL local area network (National Physical Lab, UK). A major drawback of ABP is that it only allows one message in transit on a link.

In addition, the only processes considered are *S* and *D*. The devices connected to current networks and the networks themselves are multiplexed. Fair resource sharing is imperative. The exclusive appropriation (source of waste) of shared resources is banned. It is avoided by introducing at *S* a wait between two consecutive sending of copies. But how can a process *wait for some bounded time without using a timer*?

b) Bounded Retransmission Protocol

Its aim is to transform ABP into an “efficient” transport protocol, by adding the sliding window mechanism. This protocol would work without a timer. After sending, *S* “hands over” to the local operating system (scheduler and process manager), which “gives back” control to *S* after having executed one or more processes. *S* then ends its silence (by sending one or more PDUs). And so on. Ditto for *D*.

This “solution” is based de facto on clocks and timers, utilized by operating systems! Any waiting mechanism based on executions (resource occupations in general) boils down to a "poor man's clock", time being counted or "consumed" very roughly. In Computer Science, state transitions are used to guarantee "progression" of processes — the properties of *liveness* and *termination*. Any state transition is "good to take". As it turns out, clock increments are the most precise and exact state transitions imagined so far. In addition, a clock (which enables building timers) has the advantage of freeing up useful resources wasted by unnecessary active waits.

c) Shared knowledge of RTT

S and *D* must share the same knowledge of *RTT* when opening *LV*. It can be a standard value, or it can be negotiated. The maximum durations Δ_S and Δ_D each depend on a set of parameters specific to their devices (sequential or parallel executable processes — possibly multi-threaded, scheduling algorithms based on deadlines or on fixed priorities, durations of worst-case process activation and execution, etc.).

For knowing Δ_S and Δ_D, one must conduct a schedulability analysis specific to each device. These analyses are of considerable complexity. In addition, the worst-case execution times of a process can depend on the values of input variables, suspension(s) during execution (e.g., waiting on a semaphore for entry in critical section), interrupts on external events. Therefore, in general, one considers (pessimistic) upper ceilings of worst-case durations for Δ_S and Δ_D. Δ_R depends on network loads due to traffic carried on *all* virtual links.

In addition to their relative inaccuracies and variability over time, these durations may be incompatible with the qualities of service imposed or promised by providers of access to *R*. Timers which trigger interruptions guarantee that any device will be able to ensure a given *RTT* (standard or negotiated). This is not the case in the absence of timers.

d) Risks of failure and endless waiting

Without the use of timers, an opening of *LV* can fail (delays experienced appear “too long”). More to the point: without a timer, how can *S* or *D* know whether to keep waiting or give up? Not being able to know if a silence is "abnormally long" (no PDU or ack/nack), they have to rely on a process manager which will make, on their behalf and thanks to timers, the

difference between "everything is fine, wait" and "an overload or failure has occurred, stop waiting"— *LV* is closed in the latter case.

Assertion A can therefore be rewritten as follows: **Any protocol intended to guarantee Ψ is necessarily based on shared knowledge of a finite upper bound for transport-level delays.**

An early proof showing that assertion A is correct, as well as the conditions that *RTT* must satisfy, are given in [9].

2.2. Discussion from the perspective of "distributed algorithms"

The two-process agreement problem (property Ψ) is a special case of the problems posed and studied by the "Distributed Algorithms" community from the end of the 1970s. Networks such as *R* belong to the category of asynchronous systems: delays are finite unbounded, or they are finite and bounded, but the values of the bounds are unknown (no *RTT*). There are many impossibility results. Here are two relevant examples.

a) Reliable FIFO communications

Guaranteeing Ψ amounts to proving that reliable FIFO communications are possible over networks such as *R*. It has been shown that this problem has no asynchronous solution [10]. Concerning ABP in particular:

"*Three kinds of faults are of interest when discussing reliable FIFO channels in asynchronous systems: loss, reordering, and duplication of packets. There is a solution, the Alternating Bit protocol, for the case of both loss and duplication faults. By way of contrast, no solution is possible for the case of both reordering and duplication faults*." [11]

b) Distributed Consensus

Instead of two processes (*S* and *D*), one generalizes to *n* processes, $n > 2$. For example, one considers *h* sources supposed to emit the ordered sequence Ω and *k* destinations, $h > 1$, $k > 1$ (case of highly available systems). The agreement property Ψ implies the possibility of deciding collectively whether or not a new message *X* can be added to the prefix of Ω in progress in each of the processes.

A well-known impossibility result in Computer Science was established in 1985: in the presence of a single fault (process failure), if the value of a finite upper bound of network delays is unknown, Distributed Consensus is impossible [12].

Distributed Consensus is property Ψ in the general case ($n > 2$). This result was later extended to other faults (such as message losses). See [13] for an example.

Conversely, **it is possible to prove the agreement property Ψ in *Timed* asynchronous systems** [14], which are **asynchronous systems "augmented" with the knowledge of the values of finite delay bounds (such as *RTT* in the case of Internet).**

Conclusion: Assertion A is correct

3. What now?

Thanks to technological progress (submarine optical cables at several hundred terabits/s, multicore processors at several tens of terabits/s, etc.), the processing and storage capacities within IP/MPLS network architectures (virtualized and segmented), above networks (*cloud computing*) and at the edge of networks (*mobile edge computing*) are becoming virtually unlimited. Advances based on computational power-hungry algorithmic innovations are now available (elliptic codes and authentication certificates, dynamic generation of tamper-proof

single-use passwords, unsupervised algorithmic learning, etc.). Well-known solutions (*pseudonymization*, *concurrency control*, *blockchains*, etc.) are extensions of distributed algorithms, intended to guarantee different variants of the agreement property Ψ, involving several ($h + k$) stakeholders (h and k possibly unknown):

- In explicit cooperation (production and registration of pseudonyms without violation of anonymity, multi-copied data structures in SDNs (*software defined networks*) and those used by search engines, etc.),

- In full competition (purchases-sales / auctions, financial transactions, etc.).

In the vast universe of digital technology, the transition from TCP toward distributed algorithms that began at the end of the 1970s [15] has been among the most fundamental. Understanding that transition allows us to follow the evolution of knowledge and innovations up to the present day.

With the Internet, humans began their "migration" to a cyber-physical universe, which is under construction. This is how the "symbiosis" envisioned in 1960 by Joseph Licklider, an extraordinary visionary, was realized [16]. Thanks to digital tools (voice-activated assistants, PCs and smartphones, connected bracelets, etc.) and services accessible via the Web, humans have acquired new possibilities offered by cyberspace. In doing so, they expose themselves to new dangers, including cyber espionage, theft of personal data, and digital “deaths” through identity theft or ransomware. In the not too distant future, “immersed” in communicating terrestrial and aerial vehicles, partially or fully automated, they will also be exposed to potentially lethal cyberattacks (physical deaths due to collisions).

Networks spontaneously formed by such vehicles are a particularly interesting example of mobile communicating process systems. Certain technologies at the heart of the Internet of the future will be employed (for example, compact 5G MIMO short range on-board antennas, Artificial Intelligence).

Other innovations are expected such as, e.g., *short-lived privacy-preserving naming*, *proof-of-presence*, *self-aware computing*. It is necessary to demonstrate that these networks have four fundamental properties: *safety*, *privacy*, *efficiency*, *cybersecurity*. This condition is essential in any motorized society concerned with ethics and respect for human life.

The key question then is: how to design in order to be able to prove?

◈ ◈ ◈ ◈ ◈

Glossary

ABP	*Alternating Bit Protocol*
ack	*acknowledgement*
ARQ	*Automatic Repeat request*
FIFO	*First In First Out*
IETF	*Internet Engineering Task Force*
INRIA	*Institut National de Recherche en Informatique et Automatique*
IP	*Internet Protocol*
MIMO	*Multiple-Input Multiple-Output*
MPLS	*MultiProtocol Label Switching*
nack	*negative ack*

OSI/ISO	*Open Systems Interconnection/International Standards Organisation*
PDU	*Protocol Data Unit*
RFC	*Request for Comments*
RTT	*Round Trip Time*
TCP	*Transmission Control Protocol*

Bibliography

[1] The concepts of unreliable network ("best effort") and "universal message block" (later renamed packet/datagram) were invented by Paul Baran (Rand Corporation) in 1960 and 1961.

[2] Founding publication: V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol Com-22 (5), May 1974. https://www.cs.princeton.edu/courses/archive/fall06/cos561/papers/cerf74.pdf

[3] Packets/datagrams were not sufficient.

[4] Cubic for Fast Long-Distance Networks, IETF RfC 8312, Feb. 2018, 19 p. https://datatracker.ietf.org/doc/html/rfc8312

[5] The first analytical tools making it possible to calculate Δ_R were introduced by Leonard Kleinrock in his 1962 PhD thesis (MIT).

[6] IETF RfC 6298: "Computing TCP's Retransmission Timer", June 2011. https://datatracker.ietf.org/doc/html/rfc6298

[7] Stop-and-Wait, Go-Back-N, Selective Repeat.

[8] K. Bartlett, R. Scantlebury, and P. Wilkinson, "A note on reliable full-duplex transmission over half-duplex links", Communications of the ACM, Vol. 12 (5), May 1969, pp. 260-261.

[9] G. Le Lann and H. Le Goff, "Verification and evaluation of communication protocols", Computer Networks, Vol. 2 (1), North-Holland / Elsevier, Feb. 1978, pp. 50-69. https://www.sciencedirect.com/science/article/abs/pii/0376507578900399

[10] Y. Afek et al., Reliable Communication Over Unreliable Channels, Journal of the ACM, Vol.41 (6), Nov. 1994, pp. 1267-1297.

https://groups.csail.mit.edu/tds/papers/Lynch/jacm94.pdf

[11] D. Wang and L. Zuck, "Tight bounds for the sequence transmission problem", Proceedings of 8th ACM PODC Symposium, 1989, pp. 73–83.

[12] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process", Journal of the ACM, Vol. 32 (2), April 1985, pp. 374-382. http://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf

[13] U. Schmid, B. Weiss, and I. Keidar, "Impossibility results and lower bounds for consensus under link failures", SIAM Journal of Computing, Vol. 38 (5), Jan. 2009, 1912-1951.

[14] F. Cristian and C. Fetzer, "The Timed Asynchronous Distributed System Model", IEEE Transactions on Parallel and Distributed Systems, vol. 10 (6), June 1999, pp. 642-657.

[15] G. Le Lann, "Distributed Systems - Towards a Formal Approach", Proceedings of the 7th IFIP Congress, Aug. 1977. North-Holland, ISBN 0-7204-0755-9, 155-160.

https://www.researchgate.net/publication/221329975_Distributed_Systems_Towards_a_Formal_Approach

[16] JCR Licklider , “Man-Computer Symbiosis”, IRE Transactions on Human Factors in Electronics, volume HFE-1, pages 4-11, March 1960 https://history-computer.com/man-computer-symbiosis/

Short bio

Dr. Gérard Le Lann holds a M.S. in Applied Mathematics, an Engineering Degree (ENSEEIHT, Toulouse), and a Ph.D in Computer Science (University of Rennes, France).

During his extended national service in the French Navy (1966-1969), he was involved in the design of a 4-computer real-time system intended for the first French nuclear submarine. He started his professional career at CERN, Geneva (Switzerland) in 1969, as a member of the team in charge of designing an 8-node local area network aimed at collecting and preprocessing data generated on the fly by the beam inside the particle accelerator. He joined IRIA (now INRIA) as a researcher in 1972 and a member of the Cyclades project.

At Stanford University (1973-74), working with Professor Vint Cerf on the sliding window scheme, he was involved in the design of what became known as the Internet TCP/IP protocol.

Back to INRIA, he published one of the founding papers on distributed fault-tolerant computing (1977). Appointed research director in 1978, he created and led a number of project/teams focused on nascent topics such as:

- Distributed algorithms: terminating distributed consensus, non-blocking concurrency control in distributed databases, atomic commit, and leader election, in various timing models (from full synchrony to full asynchrony),

- Deterministic multiaccess protocols for local area networks aimed at real-time applications,

- Proof-based system engineering for safety-critical systems (designs and failure analyses).

His work has resulted in more than a hundred scientific publications, contracts, patents, and industrial transfers (Digital Equipment Corp/USA, French Navy, Arianespace, etc.). G. Le Lann has also been INRIA’s representative and scientific advisor to a number of organizations.

In 2012, he has received the Willis Lamb Prize from the French Academy of Sciences for his work on distributed fault-tolerant systems aimed at critical environments (civil and defense domains).

As a research director emeritus with INRIA since 2008, he currently works on life-critical cyberphysical systems, as well as on networks of fully automated communicating vehicles, in partnership with a North American company. Safety, privacy, efficiency and cybersecurity properties are being investigated, together with ethical, juridical and societal issues.

https://www.linkedin.com/in/gerard-le-lann-85b0837/

https://www.researchgate.net/profile/Gerard-Le-Lann