The Distributed Computing Scene Seen From La Seine

Gérard Le Lann
INRIA Rocquencourt, France
Gerard.Le_Lann@inria.fr

Abstract

In this short position paper, I summarize my vision of how the past 25 years of research in distributed computing were started in the early 70's, and then developed, having spent the first part of this quarter century in the USA, and the other part in France. In addition to the names that appear in this short summary, there are many other names of great scientists I have been interacting with, which I would have liked to mention. May they be assured that they are not forgotten.

1. The past and the present

I became addicted to research work in distributed computing when at Stanford University, 1973-1974.

At that time, distributed computing meant "computer networking", i.e. explicit process/processor communications and interactions only. In the Digital Systems Laboratory, under the lead of Professor Vint Cerf, we were having fun with this new thing called the Arpanet.

As a contributor to the design of what became known as the TCP protocol, I realized how "risky" it is to use physical time for enforcing logical safety properties in distributed systems. The sliding window protocol, at the core of TCP, was one of the early algorithms solving a distributed synchronization problem in a partially synchronous computational model.

At about the same time, not too far away from the Stanford campus, Xerox Park was prototyping a revolutionary local area networking technology, the Ethernet, solving a distributed unitary (a single resource) concurrency control problem out of a mutual exclusion algorithm based upon probabilistic tree search, also considering a partially synchronous computational model. As convincing as Bob Metcalfe could be regarding the virtues of Ethernet, I resisted the temptation of taking advantage of a "magic" hardware-provided mechanism to solve "high-level" distributed computing problems. Tolerance to failures was also a concern.

A more fundamental and disturbing question was a real challenge in the mid 70's: What makes a system "distributed" (rather than "centralized")? Gradually, it became clear that the very distinguishing feature of "distribution" is the impossibility of assuming "natural" or cost-free knowledge of global system states.

The inception of the virtual ring & circulating token algorithm (1977) resulted from these considerations. Shortly after, the organizers of the 1st ICDCS Conference, Doug Jensen in particular, invited me to be the International Liaison Chairman. Huntsville, Alabama, 1979: the starting point of a very successful Conference series, which was taken to some unexpected "heights" in 1990 in Paris, when ICDCS was organized by INRIA, with the cocktail party given at the Eiffel Tower!

In the late 70's—early 80's, the writings which had the strongest influence on my research work were those from Leslie Lamport and Nancy Lynch, pioneers in the area of distributed fault-tolerant algorithms, from Robert Thomas (Bolt Beranek & Newman), Philip Bernstein and Nathan Goodman (Computer Corp. of America), pioneers in the area of distributed and replicated databases.

Along with the well known papers from L. Lamport, from N. Lynch, on clocks, event orderings, time bounds in the presence of uncertainty, mutual exclusion, impossibility proofs, I consider the 1987 book by P. Bernstein, V. Hadzilacos and N. Goodman on concurrency control and the serializability theory as one of the major milestones in the history of distributed computing.

Another major piece of work which I found very impressive in the early 80's, not very much publicized in the distributed computing scientific community, is the design of the GPS system, the first man-made (distributed) system designed after the principles of relativistic physics.

Since the late 80's, we have witnessed a flurry of results regarding optimality and impossibility proofs for generic problems in distributed fault-tolerance, for various computational models, ranging from pure synchrony to pure asynchrony.

Group membership, leader election, exact consensus, approximate agreement, are well known examples of such generic problems. I personally regard as an essential research axis the quest for computational models that are "as close as possible" to pure asynchrony, for obvious theoretical reasons, but also for very pragmatic reasons, since the coverage of systems—i.e. how "trustable" systems are when being in use—increases with the inverse of the distance between a system design model and the pure asynchronous model, for given performance and efficiency figures.

A very significant step in that direction was made by Tushar Chandra and Sam Toueg, when they published their work on unreliable failure detectors, in the mid 90's. They were the first researchers who saw how to augment the pure asynchronous model of computation with some time-free semantics sufficient for circumventing the famous Fisher-Lynch-Paterson impossibility result (1983).

Since then, an impressive number of results have been established with and for distributed algorithms (e.g., consensus, atomic broadcast, atomic commit), for various failure semantics, ranging from the "least aggressive" (permanent crash) to the "most aggressive" (transient unauthenticated Byzantine), based upon failure detectors or designed for various partially synchronous models of computation.

A very appealing feature of failure detectors is their natural potential for fitting "low" levels of abstraction, making it possible to run distributed algorithms parallel to failure detectors, which lends itself well to leveraging standard results in scheduling theory, leading to new (and improved) optimal worst-case complexity bounds, as shown in our joint work (with M. Aguilera and S. Toueg, 2000) on fast failure detectors.

That a distributed system is not "flat", that schedulers used at various abstraction/implementation levels may "favor" some processes/messages against others, and that this results into different end-to-end delays for different levels, is something not very well exploited yet.

Since the late 90's, we have been witnessing the emergence of time-free semantics (aimed at augmenting the pure asynchronous model) based upon "relativistic" assumptions regarding end-to-end message passing and processing delays, rather than absolute lower/upper delay bounds. Our current work (with U. Schmid, Vienna University of Technology) is an example of such "relativistic" approaches.

The main motivation for these types of time-free semantics lies with their ability to maintain logical safety properties in the presence of violations of postulated upper bounds on delays.

It is speculated that these types of semantics will flourish, since they match quite well the uncertainty which is germane to many distributed system settings, such as, e.g., continent-wide dispersion of publicly accessed services, peer-to-peer applications, highly mobile and wireless systems, groups of anonymous members, deep space missions, safety/life-critical embedded applications and systems.

Besides the above mentioned topics, a number of other important novel "branches" of distributed computing have emerged over the past 25 years, such as linearizability, self-stabilization, routing in ad hoc wireless networks, languages (e.g., TLA, I/O automata) and tools for formal specifications and formal validation/verification, to name a few.

2. The future

Looking backwards helps in identifying what we should do better in the future.

It appears there is a need for the distributed computing community to be more proactive in explaining and transferring the theoretical and applied research results to the real world.

How many failures of deployed distributed systems are due to design faults that could be avoided if the basic state-of-the-art in distributed computing would be more widespread and practiced than it currently is?

There seems to be two major tracks for the future of "classical" distributed computing, which are as follows:

-- A scientific track, where the many open problems that belong to the intersecting set of real-time computing (scheduling theory, queuing theory) and fault-tolerant distributed computing would be addressed. Complexity bounds, optimality proofs, proofs of timeliness, in the presence of concurrency, mobility, and failures, are not very common yet. That scheduling theory may influence the way problems in distributed computing are specified and resolved—and vice-versa—is not sufficiently well understood. There is not even an agreement on whether timeliness proofs (e.g., bounds on response times) boil down to logical safety proofs (a viewpoint backed by some fraction of the distributed computing community) or whether they can only result from solving problems in combinatorial analysis (a viewpoint backed the real-time computing community). How useful is it to mix up “real-time” with “synchrony assumptions”? From a more general standpoint, there is a need for considering generic *integrated* problems in distributed computing, not just one problem *in isolation* from others. How useful is it to ignore concurrency and failures, when addressing a distributed scheduling problem? How useful is it to claim having a solution for a distributed real-time computing problem while ignoring the existence of waiting queue phenomena?

-- An engineering track, where new system engineering lifecycles & processes based upon continuous chains of proofs, from application requirements capture to system design & validation, down to implementation, integrated testing and fielding, would be defined, increasing significantly the applicability of scientific achievements.

The future of “non classical” distributed computing is harder to predict. There are well known trends, such as, e.g., non von Neumann computers, bio-computing, fully autonomous systems. Given its wide scope, it is much more difficult to state a grand challenge for distributed computing than it is for, e.g., formal software methods. What could be, in our area, the counterpart of “The Verifying Compiler: A Grand Challenge for Computing Research” (≈ 1000 man-years of research effort, spread over 10 years or more), as defined by Tony Hoare, Journal of the ACM, January 2003?

However, given the impressive increase of the number of papers submitted to the ICDCS Conferences, we should feel confident that the future of the distributed computing field, be it the “non classical” or the “classical” type, is quite bright.