

Rediscovering Distributed Systems

Steve Vinoski • Basho Technologies



Twenty-five years ago, around the end of the 1980s, focusing on distributed systems as a researcher or practitioner meant you were part of a relatively small community of specialists. By then, computers and computing networks had evolved significantly; corporate LANs were becoming more common, Ethernet and TCP/IP were already advancing toward ubiquity, the Internet was growing rapidly, and the World Wide Web was just being invented. But distributed systems software was still finding its way, with the relatively few industrial and research teams involved each writing their own proprietary distributed software stacks, including, in many cases, entire distributed operating systems.

Shortly thereafter, the 1990s saw upheaval in the distributed systems arena. The distributed systems silos of the 1980s gave way to industry standards built, for better or worse, on distributed object architectures. At the same time, the popularity of message queuing systems grew as enterprises deployed more systems and applications requiring integration. The foremost force in this upheaval, though, was the rapid growth of the Web, which quickly reached levels of scale the distributed systems community had previously only dreamed about.

While the Web drove the deployment of larger and more numerous distributed systems toward the end of the '90s and into the new millennium, it was uncertain how much influence the previous decades of work were having on those systems' designs. It's certain that developers not involved in the distributed systems community helped design and build some of the first larger websites, and it's unclear how well versed they were in important distributed systems topics such as consensus, fault tolerance, concurrency, and availability.

But that uncertainty didn't last long. Distributed systems have continued to grow in scale

and scope since then, and today such systems no longer occupy a niche but instead impact a large segment of developers and researchers. Distributed systems provide the underpinnings for applications in various domains – such as social, financial, mobile, educational, and entertainment – so it isn't unreasonable to say that most researchers and developers are now, to some degree, involved in distributed systems. For example, applications involving a Web browser talking to a website or a cache operating in front of a server or database are common, and both are distributed systems.

Thankfully, today's newcomers to distributed systems programming are showing an interest in rediscovering the knowledge, wisdom, and lessons of past research and development. For a variety of reasons, including partial failure, latency, and concurrency, distributed systems can be incredibly difficult to reason about, implement, and debug. When faced with these and other hard problems – together with ever-increasing system scale and performance requirements – many developers eventually realize that it's a losing proposition to try to reinvent the distributed systems wheel on their own.

What if you're new to the field, or generally need to strengthen your distributed systems knowledge? The body of work produced over decades of research and development is incredibly rich, so it can be daunting to figure out what parts are still important, what parts are interesting but just historical footnotes, and what parts you can simply ignore. What follows is a brief list of publications and resources that, in my opinion, are useful to distributed systems newcomers and veterans alike.

Causality

When a system receives a message and acts on it, the system state is affected. A particular sequence

of messages or events might move the system from state S1 to state S2, whereas the same messages or events but in a different order might instead move the system to a completely different state S3. If a system is replicated for purposes of fault tolerance or availability, different replicas seeing different event ordering could well result in inconsistency across the distributed system. Many publications over the years have addressed the problem of ordering events across distributed systems, and the most cited of these is likely Leslie Lamport's "Time, Clocks, and the Ordering of Events in a Distributed System," published in 1978.¹ It explains event ordering in terms of causality as well as the proper handling of logical clocks and physical clocks across a computing cluster to address both partial ordering and total ordering of events. In 1988, Colin Fidge² developed vector clocks, which improve on Lamport's ordering approach through enhanced causality tracking but at the cost of maintaining more information about event participants and their logical clocks. (Friedemann Mattern independently discovered vector clocks around the same time as Fidge, but I find Fidge's paper easier to read.) Nuno Preguiça and colleagues created dotted version vectors in 2010,³ which further improve on earlier causality tracking techniques in several ways, such as by having the ability to capture causality more completely, and also by bounding the amount of information required to the order of degree of replication rather than the number of participants.

Consensus

If some of their parts are faulty or even intentionally malicious, distributed systems can still be reliable if they use fault-tolerant consensus protocols to reach agreement among majorities of their processes. Such protocols, though, are notoriously difficult to design and implement correctly. Consequently, numerous

papers, articles, and books, written by some of the brightest minds in the distributed systems community, are devoted to trying to solve the consensus problem. Lamport has also been quite influential in this area, as with his Byzantine Generals paper from 1982,⁴ which showed how systems can achieve consensus even in the presence of malicious processes. Around the same time, Michael Fischer, Nancy Lynch, and Michael Paterson published their famous "FLP Impossibility" paper,⁵ which proved that an asynchronous system can't achieve consensus in bounded time in the presence of just a single faulty process. (As Nancy Lynch later explained in a talk she gave in 1989, such proofs are important for knowing "when you should stop trying to devise or improve an algorithm.")⁶

Around 1990, Lamport submitted a paper documenting Paxos, which today is arguably the most well-known consensus protocol, but reviewers panned his paper because of its unusual metaphorical style. Regardless, Paxos flourished – for example, in 1996, Butler Lampson published a paper about using Paxos to build highly available systems⁷ – and so Lamport's paper was finally published in 1998.⁸ Since then, hundreds of papers about consensus in general and Paxos in particular have appeared, in part because Paxos has a reputation for being very difficult to both understand and implement, despite Lamport's attempt to rectify the situation in his 2001 paper, "Paxos Made Simple."⁹ In 2013, Diego Ongaro and John Ousterhout published a paper about a consensus protocol called Raft, which they claim "is easier for students to learn than Paxos."¹⁰ Just the prospect of a consensus protocol easier than Paxos led quite a few developers to publish open source implementations of Raft in a variety of programming languages in 2013.

Around the same time Paxos was initially developed, Barbara Liskov and Brian Oki devised Viewstamped Replication,¹¹ similar to Paxos but with an explicit replication aspect. Its focus on high availability and not just consistency makes it a compelling consideration for today's systems. My colleague Justin Sheehy, CTO of Basho Technologies, feels that if Viewstamped Replication were more widely known, there might be less need for simpler Paxos alternatives such as Raft. You can find more details about it on its project website (www.pmg.csail.mit.edu/vr/).

Availability

Toward the end of the '90s, as websites and content-delivery networks were honing the practice of scaling horizontally across numerous commodity servers instead of scaling vertically by replacing machines with more powerful ones, Eric Brewer realized there was a fundamental tradeoff at play in such systems. He explained this tradeoff – which would come to be known as the CAP Theorem – in his keynote at the 2000 ACM Symposium on Principles of Distributed Computing.¹² His insight was that in a distributed system, where machine and network failures are a matter of course and so must be accommodated, achieving both full consistency and complete availability is impossible. Around the same time, Brewer and coauthor Armando Fox used the notions of "harvest" and "yield" to clearly explain the tradeoffs of the CAP Theorem for practical applications.¹³ The CAP Theorem, formally proven in 2002, has since spawned numerous articles, papers, and blog posts, many of them seeming to misunderstand the theorem, and a few even erroneously claiming to have "beaten the CAP Theorem." In a follow-up paper in 2012, Brewer took a look back at CAP and how it applies to today's systems, and cleared up some of the confusion surrounding it.¹⁴

Unfortunately, what I've listed here barely scratches the surface; there are far more categories and publications I could include given more publication space. For more information, please see the following resources:

- Two of my Basho Technologies colleagues maintain online distributed systems lists: Chris Meiklejohn maintains a reading list (<http://bit.ly/196FF3R>), and John Daily keeps a list of distributed systems reading lists and resources (<https://gist.github.com/macintux/6227368>).
- Michael Bernstein gave a wonderful talk at the RICON West conference in October 2013 entitled "Distributed Systems Archeology," in which he explored distributed systems issues and some seminal papers that address them.

Both his presentation slides and the video of his talk are available online (<http://bit.ly/11JdhF3>).

- At the GOTO Berlin conference in October 2013, I gave a talk entitled "Rediscovering Distributed Systems" that covered several important distributed systems papers from the 1960s to the present. My presentation slides, which contain seven slides of references, are available online (<http://bit.ly/1az5Knc>).

I recommend not only reading and studying the papers mentioned here and in these resources, but also following the references each paper cites and exploring those as well. Doing so will help you develop a deeper understanding of distributed systems and why they continue to present some of the most difficult

problems and challenges the field of computing has to offer. □

References

1. L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Comm. ACM*, vol. 21, no. 7, 1978, pp. 558–565.
2. C. Fidge, "Timestamps in Message-Passing Systems That Preserve the Partial Ordering," *Australian Computer Science Comm.*, vol. 10, no. 1, 1988, pp. 56–66.
3. N. Preguiça et al., "Dotted Version Vectors: Logical Clocks for Optimistic Replication," Computing Research Repository, submitted for publication, 2010; <http://arxiv.org/abs/1011.5808>.
4. L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems*, vol. 4, no. 3, 1982, pp. 382–401.
5. M.J. Fischer, N.A. Lynch, and M.S. Paterson, "Impossibility of Distributed

IEEE Internet Computing

Editor in Chief

Michael Rabinovich • michael.rabinovich@case.edu

Associate Editors in Chief

M. Brian Blake • m.brian.blake@miami.edu
 Barry Leiba • barryleiba@computer.org
 Maarten van Steen • steen@cs.vu.nl

Editorial Board

Virgilio Almeida • virgilio@dcc.ufmg.br
 Elisa Bertino • bertino@cerias.purdue.edu
 Fabian Bustamante • fabianb@cs.northwestern.edu
 Yih-Farn Robin Chen • chen@research.att.com
 Vinton G. Cerf • vint@google.com
 Siobhán Clarke • siobhan.clarke@cs.tcd.ie
 Fred Douglass • f.douglass@computer.org
 Schahram Dustdar • dustdar@dsg.tuwien.ac.at
 Stephen Farrell • stephen.farrell@cs.tcd.ie
 Robert E. Filman* • filman@computer.org
 Carole Goble • cag@cs.man.ac.uk
 Michael N. Huhns • huhns@sc.edu
 Anne-Marie Kermarrec • anne-marie.kermarrec@inria.fr
 Anirban Mahanti • anirban.mahanti@nicta.com.au
 Cecilia Mascolo • cecilia.mascolo@cl.cam.ac.uk
 Peter Mika • pmika@yahoo-inc.com
 Dejan Milojčić • dejan@hpl.hp.com
 George Pallis • gpallis@cs.ucy.ac.cy
 Charles J. Petrie* • petrie@stanford.edu
 Gustavo Rossi • gustavo@lifia.info.unlp.edu.ar
 Amit Sheth • amit.sheth@wright.edu

Munindar P. Singh* • singh@ncsu.edu
 Torsten Suel • suel@poly.edu
 Doug Tygar • tygar@cs.berkeley.edu
 Steve Vinoski • vinoski@ieee.org
 * EIC emeritus

CS Magazine Operations Committee

Paolo Montuschi (chair), Erik R. Altman, Nigel Davies, Lars Heide, Simon Liu, Cecilia Metra, Shari Lawrence Pfleger, Michael Rabinovich, Forrest Shull, John R. Smith, Gabriel Taubin, George K. Thiruvathukal, Ron Vetter, and Daniel Zeng

CS Publications Board

Thomas M. Conte (chair), Alain April, David Bader, Angela R. Burgess, Greg Byrd, Koen DeBosschere, Frank E. Ferrante, Paolo Montuschi, Linda I. Shafer, and Per Stenström

Staff

Editorial Management: Rebecca Deuel-Gallegos
 Lead Editor: Brian Kirk, bkirk@computer.org
 Publications Coordinator: internet@computer.org
 Contributors: Keri Schreiner, Nancy Talbert, and Joan Taylor
 Director, Products & Services: Evan Butterfield
 Senior Manager, Editorial Services: Robin Baldwin
 Senior Business Development Manager: Sandy Brown
 Membership Development Manager: Cecelia Huffman
 Senior Advertising Supervisor: Marian Anderson, manderson@computer.org

Technical cosponsor:



IEEE Internet Computing
 IEEE Computer Society Publications Office
 10662 Los Vaqueros Circle
 Los Alamitos, CA 90720 USA

Editorial. Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Internet Computing* does not necessarily constitute endorsement by IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length.

Submissions. For detailed instructions, see the author guidelines (www.computer.org/internet/author.htm) or log onto *IEEE Internet Computing's* author center at ScholarOne (<https://mc.manuscriptcentral.com/cs-ieee>). Articles are peer reviewed for technical merit.

Letters to the Editors. Email lead editor Linda World, lworld@computer.org

On the Web. www.computer.org/internet/

Subscribe. Visit www.computer.org/subscribe/

Subscription Change of Address. Send requests to address.change@ieee.org

Missing or Damaged Copies. Contact help@computer.org

To Order Article Reprints. Email internet@computer.org or fax +1 714 821 4010.

IEEE prohibits discrimination, harassment, and bullying. For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html

- Consensus with One Faulty Process," *Proc. 2nd ACM SIGACT-SIGMOD Symp. Principles of Database Systems* (PODS 83), 1983, pp. 1–7.
6. N. Lynch, "A Hundred Impossibility Proofs for Distributed Computing," *Proc. 8th Ann. ACM Symp. Principles of Distributed Computing* (PODC 89), 1989, pp. 1–28.
7. B.W. Lampson, "How to Build a Highly Available System Using Consensus," *Proc. 10th Int'l Workshop Distributed Algorithms* (WDAG 96), 1996, pp. 1–17.
8. L. Lamport, "The Part-Time Parliament," *ACM Trans. Computing Systems*, vol. 16, no. 2, 1998, pp. 133–169.
9. L. Lamport, "Paxos Made Simple," *ACM SIGACT News*, Dec. 2001, pp. 51–58; <http://research.microsoft.com/en-us/um/people/lamport/pubs/paxos-simple.pdf>.
10. D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," 2013; <https://ramcloud.stanford.edu/wiki/download/attachments/11370504/raft.pdf>.
11. B.M. Oki and B.H. Liskov, "Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems," *Proc. 7th Ann. ACM Symp. Principles of Distributed Computing* (PODC 88), 1988, pp. 8–17.
12. E.A. Brewer, "Towards Robust Distributed Systems," *Proc. 19th Ann. ACM Symp. Principles of Distributed Computing* (PODC 00), 2000; www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.
13. A. Fox and E.A. Brewer, "Harvest, Yield, and Scalable Tolerant Systems," *Proc. 7th Workshop Hot Topics in Operating Systems* (HOTOS 99), 1999, pp. 174–178.
14. E. Brewer, "CAP Twelve Years Later: How the 'Rules' Have Changed," *Computer*, vol. 45, no. 2, 2012, pp. 23–29.

Steve Vinoski is an architect at Basho Technologies in Cambridge, Massachusetts, and a former columnist for *IEEE Internet Computing*. He's a senior member of IEEE and a member of ACM. You can read Vinoski's blog at <http://steve.vinoski.net/blog/> and contact him at vinoski@ieee.org or on Twitter at [@stevevinoski](https://twitter.com/stevevinoski).

cn Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

IEEE STC 2014

26th Annual IEEE Software Technology Conference



March 29-April 3, 2014
Long Beach, CA, USA

"Meeting Real World Challenges through Software Technology" is the theme of STC 2014. As technologists and as citizens, we are faced with a myriad of challenges from defending national security, to ensuring the robustness of our critical infrastructure, to sustaining and enhancing large portfolios of legacy systems – all within ever tighter resource constraints. Many of our attendees and their customers, rather than creating brand new software-intensive systems, will be updating code in embedded systems, integrating new capabilities, or otherwise retrofitting existing deployed systems.

<http://ieee-stc.org/>

Register today!



IEEE  computer society