Brief Announcement: What's Live? Understanding Distributed Consensus

Saksham Chand*
schand@cs.stonybrook.edu
Computer Science Department, Stony Brook University
Stony Brook, New York, USA

Yanhong A. Liu
liu@cs.stonybrook.edu
Computer Science Department, Stony Brook University
Stony Brook, New York, USA

ABSTRACT

Distributed consensus algorithms such as Paxos have been studied extensively. Many different liveness properties and assumptions have been stated for them, but there are no systematic comparisons for better understanding of these properties.

This paper systematically studies and compares different liveness properties stated for over 30 prominent consensus algorithms and variants. We introduced a precise high-level language and formally specified these properties in the language. We then create a hierarchy of liveness properties combining two hierarchies of the assumptions used and a hierarchy of the assertions made, and compare the strengths and weaknesses of algorithms that ensure these properties. Our formal specifications and systematic comparisons led to the discovery of a range of problems in various stated liveness properties. We also developed TLA+ specifications of these liveness properties, and we used model checking of execution steps to illustrate liveness patterns for Paxos.

CCS CONCEPTS

• **Computing methodologies → Distributed computing methodologies**.

KEYWORDS

distributed consensus, liveness

ACM Reference Format:
Saksham Chand and Yanhong A. Liu. 2021. Brief Announcement: What's Live? Understanding Distributed Consensus. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3465084.3467947

The full version of this paper is available at [10].

1 INTRODUCTION

Distributed consensus is a fundamental problem in distributed systems. Many algorithms and variations have been created for distributed consensus. These algorithms are required to be safe in that only a single value or a single sequence of values have been agreed on at any time, and the value or values agreed on are among the values proposed by the processes.

However, liveness of these algorithms—certain desired progress has been made in reaching agreement—has many variations and lacks systematic comparisons. Furthermore, it is sometimes unclear or ambiguous what exactly liveness means for these algorithms.

This paper considers the most prominent consensus algorithms and variants listed in Table 1 and presents our results from precisely specifying, comparing, and analyzing their liveness properties that are stated in the literature, including the assumptions under which the properties are satisfied. We categorize the assumptions into (1) link assumptions, i.e., what is assumed about the communication links, and (2) server assumptions, i.e., what is assumed about the servers in the system. We relate different assumptions and assertions by creating three hierarchies. Together they form an overall hierarchy of liveness properties, which includes all the liveness properties stated for the algorithms and variants considered.

We introduced a precise high-level language and formally specified all these assumptions and assertions. In fact, it was the precise language and formal specifications that allowed us to establish the precise relationships among all the assumptions and assertions. The overall hierarchy not only helps in understanding liveness, but also helps in finding tighter results. It led us to the discovery of a range of problems in various stated liveness properties, from lacking assumptions or too weak assumptions under which no liveness assertions can hold, to too strong assumptions making it trivial or uninteresting to satisfy the liveness assertions. For example,

- EPaxos [36] assumes that eventually, there is always some quorum of servers that is non-faulty (what we call **Alw-Q**), but this is too weak, because each such quorum might not stay non-faulty long enough to make progress.
- Zab [20], Paxso-EPR [39], and some others assume that eventually, there is a server P and a quorum Q of servers, such that always, P is non-faulty and is the primary and Q is non-faulty (what we call **PQ-Alw**). Such strong assumptions make it trivial to reach consensus.

This also led to the question of what the right liveness properties are. In fact, many protocols assert stronger properties than necessary.

2 CONSENSUS PROBLEM AND ALGORITHMS

A distributed system is a set of processes each operating on its local data and communicating with other processes by sending and receiving messages. A process may crash and may later recover, and a message may be lost, delayed, put out of order, or duplicated. A process is said to be *non-faulty* if "it eventually performs the actions that it should, such as responding to messages" [30].

The basic consensus problem, called single-value consensus, is to ensure that a single value is chosen by non-faulty processes

*Current affiliation: Apple Inc.

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07.
https://doi.org/10.1145/3465084.3467947

from among the set of values proposed by the processes. The more general consensus problem, called multi-value consensus, is to choose a single sequence of values, instead of a single value.

We identify two types of processes: *servers* execute some consensus algorithm and together provide consensus as a service, and *clients* use this service. For multi-value consensus, clients send values to servers for them to order, and each server maintains a *log*, i.e., a sequence of chosen values, with indices called *slots*.

A server is said to be *primary* if it is explicitly elected by the algorithm to be the single primary or leader among all servers in the system. In general, the defining property of the primary is that it is the only server that can propose values to other servers. Note that some consensus algorithms do not use a primary.

A *quorum system Qs* is a set of subsets, called quorums, of servers such that any two quorums overlap. A quorum is said to be *non-faulty* if each server in it is non-faulty, and *faulty* otherwise.

Table 1 shows the algorithms and variants studied in five groups:

(1) algorithms based on Virtual Synchrony [6], the first algorithm for reliable group communication using broadcast primitives.
(2) an algorithm and variant called Viewstamped Replication [37], the earliest that has a primary-backup architecture.
(3) an algorithm and three core variants in the Paxos [28] family, the earliest algorithm based on symmetric leader election.
(4) two single-value consensus algorithms with failure detection, for crash failure model and crash-recover failure model.
(5) other algorithm variations, where the earliest (14-15) and latest (25-31) are specified formally, in languages with automated checking, and with proof support or compilation and execution.

In essence, all algorithms in Table 1 are quorum-based and run in *rounds*. A round is identified by a unique number, and is started by at most one server. Multiple rounds may run simultaneously.

- A server initiates a new round once it thinks that the current round has died, i.e., stopped making progress. Different algorithms use different conditions to decide this.
- A server *learns* a value upon either receiving *votes* from a quorum of servers for that value in some round or a message from some other server informing it about the value learned.
- If the server is maintaining a state machine, it then *executes* the learned value on the state machine.
- The *result* of executing a value is sent to the client as a *response* to the value it initially *requested*.

3 LIVENESS SPECIFICATION AND HIERARCHY

Table 2 shows the total order of link assumptions, in increasing strength from top to bottom. Figure 1 shows the hierarchies for server assumptions and liveness assertions. An arrow from node A to node B denotes that A implies B; we say that A is *stronger* than B. These assumptions and assertions are summarized below.

Link assumptions.

(1) **Raw**: messages sent may or may not be received.
(2) **Fair**: all links between servers are *fair*, that is "if a correct process repeatedly sends a message to another correct process, at least one copy is eventually delivered" [40].
(3) **Sure**: the time between a message being sent and the message being received has a known upper bound.

	Name	Description	Language used
1	VS-ISIS	Reliable group communication, Birman-Joseph 1987 [7]	English (items)
2	VS-ISIS2	Virtual synchrony, Birman-Joseph 1987 [6]	English
3	EVS	Extended virtual synchrony for network partition, Amir et al 1995 [2, 3]	pseudocode
4	Paxos-VS	Virtually synchronous Paxos, Birman-Malkhi-van Renesse 2012 [8]	pseudocode
5	Derecho	Virtually synchronous state machine replication, Jha et al 2019 [19]	pseudocode
6	VR	Viewstamped replication, Oki-Liskov 1988 [37]	pseudocode (coarse)
7	VR-Revisit	VR revisited, Liskov 2012 [32]	English (items)
8	Paxos-Synod	Paxos in part-time parliament, Lamport 1998 [28]	TLA (single-value)
9	Paxos-Basic	Single-value Paxos, Lamport 2001 [29]	English (items)
10	Paxos-Fast	Single-value Paxos with replicas proposing, Lamport 2006 [30]	English (items), TLA+
11	Paxos-Vertical	Single-value Paxos with external starting of leader election, Lamport-Malkhi-Zhou 2009 [31]	PlusCal
12	CT	Single-value consensus with crash failures, Chandra-Toueg 1996 [13]	pseudocode
13	ACT	Single-value consensus in crash-recovery model, Aguilera-Chen-Toueg 2000 [1]	pseudocode
14	Paxos-Time	Paxos with time analysis, De Prisco-Lampson-Lynch 2001 [14]	IOA (single-value)
15	Paxos-PVS	Single-value Paxos for proof, Kellomäki 2004 [25]	PVS
16	Chubby	Paxos in Google's Chubby lock service, Burrows 2006 [9]	English (partial items)
17	Chubby-Live	Chubby in Paxos made live, Chandra-Griesemer-Redstone 2007 [12]	English
18	Paxos-SB	Paxos for system builders, Kirsch-Amir 2008 [27]	pseudocode
19	Mencius	Paxos with leaders proposing in turn, Mao et al 2008 [34]	English (items)
20	Zab	Yahoo/Apache's Zookeeper atomic broadcast, Junqueira-Reed-Serafini 2011 [20]	English (items)
21	Zab-FLE	Zab with fast leader election, Medeiros 2012 [35]	pseudocode
22	EPaxos	Egalitarian Paxos, Moraru-Andersen-Kaminsky 2013 [36]	pseudocode
23	Raft	Consensus in RAMCloud, Ongaro-Ousterhout 2014 [38]	pseudocode
24	Paxos-Complex	Paxos made moderately complex, van Renesse-Altinbuken 2015 [40]	pseudocode, Python
25	IronRSL	Paxos in Microsoft's IronFleet for proof, Hawblitzel et al 2015 [18]	Dafny
26	Paxos-TLA	Paxos for proof using TLAPS, Chand-Liu-Stoller 2016 [11]	TLA+
27	LastVoting-PSync	Single-value Paxos in Heard-Of model for proof, Drăgoi-Henzinger-Zufferey 2016 [15]	PSync
28	Raft-Verdi	Raft for proof using Coq, Wilcox-Sergey-Tatlock 2017 [41]	Verdi
29	Paxos-EPR	Paxos in effectively propositional logic for proof, Padon et al 2017 [39]	Ivy
30	Paxos-Decon	Paxos deconstructed, Garcia et al 2018 [16, 17]	Scala/Akka
31	Paxos-High	Paxos in high-level executable specification, Liu-Chand-Stoller 2019 [33]	DistAlgo

Table 1: Distributed consensus algorithms and variants, and languages used to express them.

Asm	Algorithms and Variants
Raw	VS-ISIS, VS-ISIS2, Derecho, VR, VR-Revisit, Paxos-Basic, Paxos-Vertical, Paxos-PVS, Chubby, Zab-FLE, Raft, Paxos-TLA, Raft-Verdi, Paxos-Decon, Paxos-High
Fair	EVS, Paxos-VS, Paxos-Fast, CT, ACT, Paxos-SB, Mencius, EPaxos, Paxos-Complex, Paxos-EPR
Sure	Paxos-Synod, Paxos-Time, Chubby-Live, Zab, IronRSL, LastVoting-PSync

Table 2: Link assumptions used by algorithms and variants.

Server assumptions. All definitions are implicitly prefixed with "eventually".

Alw-Q: there is always some quorum of servers that is non-faulty.
Q-Alw: there is some quorum of servers that is always non-faulty.

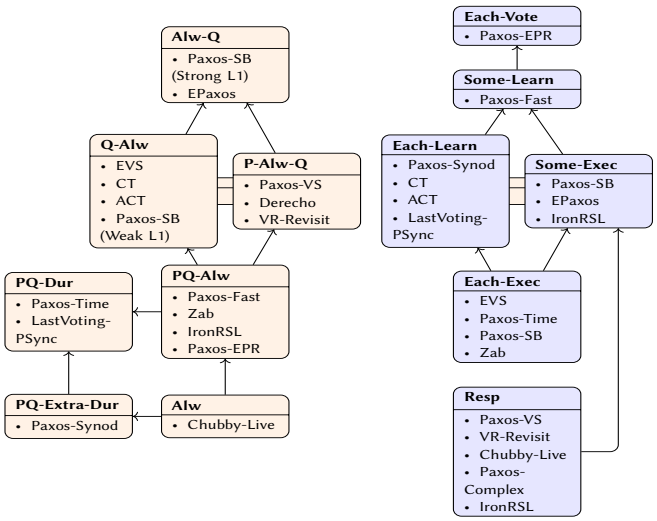

Figure 1: Hierarchies of server assumptions (left) and liveness assertions (right) as stated by the algorithms' authors.

P-Alw-Q: there is a server P such that, always, P is non-faulty and is the primary and there is some quorum Q that is non-faulty.
PQ-Alw: there is a server P and a quorum Q of servers, such that always, P is non-faulty and is the primary and Q is non-faulty.
Alw: all servers are always non-faulty, i.e., non-faulty forever from that time on.
PQ-Dur: there is a server P and a quorum Q such that P and Q are non-faulty and P is the primary, for some duration of time.
PQ-Extra-Dur: there is a duration of time $D1 + D2$ such that (1) the set of non-faulty servers does not change in the duration, (2) after duration $D1$, primary is selected, and (3) at least a fixed quorum of servers is non-faulty during the entire duration.

Liveness assertions. For brevity, we omit describing slots here. All definitions are implicitly prefixed with "eventually".

Each-Vote: in some round, each server of some quorum sends a vote for the same value in that round.
Some-Learn: some non-faulty server learns a value.
Each-Learn: each server of some quorum learns the same value.
Some-Exec: some non-faulty server executes a value.
Each-Exec: each server of some quorum executes the same value.
Resp: each client request is responded to.

4 LIVENESS PROPERTIES AND ANALYSIS

Table 3 summarizes the assumptions, assertions, the kind of proofs of the algorithms discussed, and our analysis.

Proposition 1 (**Each-Vote** insufficient). **Each-Vote** *is not strong enough as a liveness assertion.*

Proposition 2 (None from **Raw**). *No algorithm can satisfy any liveness assertion described above under* **Raw**.

Proposition 3 (All from **Q-Alw** single). *There is an algorithm that solves single-value consensus and satisfies all liveness assertions described above under* **Fair** *and* **Q-Alw**.

Proposition 4 (All from **Q-Alw**). *There is an algorithm that claims to solve consensus and satisfies all liveness properties described above under* **Fair** *and* **Q-Alw**.

	Name	Assumptions		Assertions	Proofs	Analysis
		Link	Server			
3	EVS	**Fair**	**Q-Alw**	**Each-Exec**	Systematic	(Prop 4)
4	Paxos-VS	**Fair**	**P-Alw-Q**	**Resp**	-	(Prop 5)
5	Derecho	**Fair**	**P-Alw-Q**	"progress"	-	lacking assertions
7	VR-Revisit	**Raw**	**P-Alw-Q**	**Resp**	Prose	(Prop 2)
8	Paxos-Synod	**Sure**	**PQ-Extra-Dur**	**Each-Learn**+	Prose	assuming primary
10	Paxos-Fast	**Fair**	**PQ-Alw**	**Some-Learn**	Systematic	(Cor 5.1)
12	CT	**Fair**	**Q-Alw**	**Each-Learn**	Systematic	(Prop 3)
13	ACT	**Fair**	**Q-Alw**	**Each-Learn**	Systematic	(Prop 3)
14	Paxos-Time	**Sure**	**PQ-Dur**	**Each-Exec**+	Systematic	assuming primary
17	Chubby-Live	**Sure**	**Alw**	**Resp**	-	(Cor 5.1), trivial from **Alw**
18	Paxos-SB	**Fair**	**Q-Alw**	**Some-Exec**	-	(Prop 4)
19	Mencius	**Fair**	-	"liveness"	-	lacking assumptions
20	Zab	**Sure**	**PQ-Alw**	**Each-Exec**	Sketch	(Cor 5.1)
22	EPaxos	**Fair**	**Alw-Q**	**Some-Exec**	-	(Prop 7)
24	Paxos-Complex	**Fair**	-	**Resp**	-	lacking assumptions, (Prop 6)
25	IronRSL	**Sure**	**PQ-Alw**	**Some-Exec**, **Resp**	Formal (Dafny)	(Cor 5.1)
26	LastVoting-PSync	**Sure**	**PQ-Dur**	**Each-Learn**+	Formal (PSync)	assuming primary
29	Paxos-EPR	**Fair**	**PQ-Alw**	**Each-Vote**	Formal (Ivy)	(Cor 5.1), (Prop 1)

Table 3: Assumptions, assertions, and proofs, as provided by the respective authors, together with our analysis. "-" indicates that the information is not provided. "+" in Assertions indicates that the calculation of the bound after which the assertion would be satisfied is also given.

Proposition 5 (**Resp** from **P-Alw-Q**). *There is an algorithm that claims to solve consensus and satisfy* **Resp** *under* **Fair** *and* **P-Alw-Q**.

Corollary 5.1 (Weak from **PQ-Alw**). *A liveness property that assumes* **PQ-Alw** *is a weak property, because the assumption is stronger than necessary.*

Proposition 6 (Not **Resp**). *Paxos-Complex cannot satisfy* **Resp**, *even with* **Alw**.

The following impossibility result was also discussed by Kirsch and Amir [26, Strong L1] and proved by Keidar and Shraer [23, 24].

Proposition 7 (None from **Alw-Q**). *No quorum-based consensus algorithm that executes in rounds can satisfy* **Some-Learn** *under* **Fair** *and* **Alw-Q**.

What are the right liveness properties? Clearly, **Resp** guarantees server response to client requests, and **Each-Exec** and **Each-Learn** are not needed for **Resp**. One can also see in existing algorithms that **Some-Learn** ensures consensus, while **Each-Vote** might not, and **Fair** with either **Q-Alw** or **P-Alw-Q** can ensure **Some-Learn**. Stronger assumptions with durations can yield stronger assertions.

5 RELATED WORK AND CONCLUSION

Besides the works already discussed, there are additional studies of consensus algorithms and variants, building on earlier works, relating existing algorithms, and discussing liveness, e.g., COReL [21, 22], Congruity [4, 5], and GIRAF [23, 24]. Our work is a first step in precise specification of the wide variety of liveness properties. Much future work is needed for precise complexity analysis for liveness.

ACKNOWLEDGMENTS

This work was supported in part by NSF under grants CCF-1414078 and CCF-1954837 and ONR under grant N000142012751.

REFERENCES

[1] Marcos Kawazoe Aguilera, Wei Chen, and Sam Toueg. 2000. Failure Detection and Consensus in the Crash-recovery Model. *Distrib. Comput.* 13, 2 (April 2000), 99–125. https://doi.org/10.1007/s004460050070

[2] Yair Amir. 1995. *Replication Using Group Communication Over a Partitioned Network.* Ph.D. Dissertation. Hebrew University of Jerusalem, Jerusalem, Israel. Advisor(s) Dolev, Danny. http://www.dsn.jhu.edu/~yairamir/Yair_phd.pdf

[3] Yair Amir, Louise E. Moser, Peter M. Melliar-Smith, eborah A. Agarwal, and Paul Ciarfella. 1995. The Totem Single-ring Ordering and Membership Protocol. *ACM Trans. Comput. Syst.* 13, 4 (Nov. 1995), 311–342. https://doi.org/10.1145/210223.210224

[4] Yair Amir and Ciprian Tutu. 2001. *From Total Order to Database Replication.* Technical Report. Johns Hopkins University, Baltimore, MD, USA. http://www.dsn.jhu.edu/pub/papers/cnds-2001-6.pdf CNDS-2001-6.

[5] Yair Amir and Ciprian Tutu. 2002. From Total Order to Database Replication. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS '02).* IEEE Computer Society, Los Alamitos, CA, USA, 494–503. https://doi.org/10.1109/ICDCS.2002.1022299

[6] Kenneth P. Birman and Thomas A. Joseph. 1987. Exploiting Virtual Synchrony in Distributed Systems. *SIGOPS Oper. Syst. Rev.* 21, 5 (Nov. 1987), 123–138. https://doi.org/10.1145/37499.37515

[7] Kenneth P. Birman and Thomas A. Joseph. 1987. Reliable Communication in the Presence of Failures. *ACM Trans. Comput. Syst.* 5, 1 (Jan. 1987), 47–76. https://doi.org/10.1145/7351.7478

[8] Kenneth P. Birman, Dahlia Malkhi, and Robbert Van Renesse. 2010. *Virtually Synchronous Methodology for Dynamic Service Replication.* Technical Report MSR-TR-2010-151. Microsoft Research, Redmond, WA, USA. https://www.microsoft.com/en-us/research/wp-content/uploads/2010/11/vs-submit.pdf

[9] Mike Burrows. 2006. The Chubby Lock Service for Loosely-coupled Distributed Systems. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation* (Seattle, Washington) *(OSDI '06).* USENIX Association, Berkeley, CA, USA, 335–350. http://dl.acm.org/citation.cfm?id=1298455.1298487

[10] Saksham Chand and Yanhong A. Liu. 2020. What's live? Understanding Distributed Consensus. *Computing Research Repository* arXiv:2001.04787 [cs.DC] (Jan. 2020). http://arxiv.org/abs/2001.04787

[11] Saksham Chand, Yanhong A. Liu, and Scott D. Stoller. 2016. Formal Verification of Multi-Paxos for Distributed Consensus. In *FM 2016: Formal Methods* (Limassol, Cyprus) *(FM '16).* Springer International Publishing, Cham, Switzerland, 119–136. https://doi.org/10.1007/978-3-319-48989-6_8

[12] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. 2007. Paxos Made Live: An Engineering Perspective. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing* (Portland, Oregon, USA) *(PODC '07).* ACM, New York, NY, USA, 398–407. https://doi.org/10.1145/1281100.1281103

[13] Tushar Deepak Chandra and Sam Toueg. 1996. Unreliable Failure Detectors for Reliable Distributed Systems. *J. ACM* 43, 2 (March 1996), 225–267. https://doi.org/10.1145/226643.226647

[14] Roberto De Prisco, Butler Lampson, and Nancy Lynch. 2000. Revisiting the PAXOS Algorithm. *Theor. Comput. Sci.* 243, 1-2 (July 2000), 35–91. https://doi.org/10.1016/S0304-3975(00)00042-6

[15] Cezara Drăgoi, Thomas A. Henzinger, and Damien Zufferey. 2016. PSync: A Partially Synchronous Language for Fault-tolerant Distributed Algorithms. *SIGPLAN Not.* 51, 1 (Jan. 2016), 400–415. https://doi.org/10.1145/2914770.2837650

[16] Álvaro García-Pérez, Alexey Gotsman, Yuri Meshman, and Ilya Sergey. 2018. Paxos Consensus, Deconstructed and Abstracted. In *Programming Languages and Systems* (Thessaloniki, Greece) *(ESOP '18).* Springer International Publishing, Cham, Switzerland, 912–939. https://doi.org/10.1007/978-3-319-89884-1_32

[17] Álvaro García-Pérez, Alexey Gotsman, Yuri Meshman, and Ilya Sergey. 2018. Paxos Consensus, Deconstructed and Abstracted (Extended Version). *Computing Research Repository* arXiv:1802.05969 (2018), 44 pages. https://arxiv.org/abs/1802.05969

[18] Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch, Bryan Parno, Michael L. Roberts, Srinath Setty, and Brian Zill. 2015. IronFleet: Proving Practical Distributed Systems Correct. In *Proceedings of the 25th Symposium on Operating Systems Principles* (Monterey, California) *(SOSP '15).* ACM, New York, NY, USA, 1–17. https://doi.org/10.1145/2815400.2815428

[19] Sagar Jha, Jonathan Behrens, Theo Gkountouvas, Matthew Milano, Weijia Song, Edward Tremel, Robbert Van Renesse, Sydney Zink, and Kenneth P. Birman. 2019. Derecho: Fast State Machine Replication for Cloud Services. *ACM Trans. Comput. Syst.* 36, 2, Article 4 (April 2019), 49 pages. https://doi.org/10.1145/3302258

[20] Flavio P. Junqueira, Benjamin C. Reed, and Marco Serafini. 2011. Zab: High-performance Broadcast for Primary-backup Systems. In *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN '11).* IEEE Computer Society, Washington, DC, USA, 245–256. https://doi.org/10.1109/DSN.2011.5958223

[21] Idit Keidar. 1994. *A Highly Available Paradigm for Consistent Object Replication.* Master's thesis. The Hebrew University of Jerusalem. https://www.cs.huji.ac.il/labs/transis/Abstracts/keidar-msc.html CS95-5.

[22] Idit Keidar and Danny Dolev. 1996. Efficient Message Ordering in Dynamic Networks. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing* (Philadelphia, Pennsylvania, USA) *(PODC '96).* ACM, New York, NY, USA, 68–76. https://doi.org/10.1145/248052.248062

[23] Idit Keidar and Alexander Shraer. 2006. Timeliness, Failure-detectors, and Consensus Performance. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing* (Denver, Colorado, USA) *(PODC '06).* ACM, New York, NY, USA, 169–178. https://doi.org/10.1145/1146381.1146408

[24] Idit Keidar and Alexander Shraer. 2006. *Timeliness, Failure-detectors, and Consensus Performance.* Technical Report. Technion - Israel Institute of Technology, Haifa, Israel. http://www.cs.technion.ac.il/~shralex/GIRAF-TR-Feb06.pdf CCIT 576.

[25] Pertti Kellomäki. 2004. *An annotated specification of the consensus protocol of Paxos using superposition in PVS.* Report 36. Institute of Software Systems, Tampere University of Technology, Tampere, Finland. http://www.cs.tut.fi/ohj/laitosraportit/report36-paxos.pdf

[26] Jonathan Kirsch and Yair Amir. 2008. *Paxos for system builders.* Technical Report. Johns Hopkins University, Baltimore, MD, USA. http://www.cnds.jhu.edu/pub/papers/cnds-2008-2.pdf CNDS-2008-2.

[27] Jonathan Kirsch and Yair Amir. 2008. Paxos for System Builders: An Overview. In *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware* (Yorktown Heights, New York, USA) *(LADIS '08).* ACM, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/1529974.1529979

[28] Leslie Lamport. 1998. The Part-time Parliament. *ACM Trans. Comput. Syst.* 16, 2 (May 1998), 133–169. https://doi.org/10.1145/279227.279229

[29] Leslie Lamport. 2001. Paxos made simple. *ACM SIGACT News* 32, 4 (Dec. 2001), 51–58. https://doi.org/10.1145/568425.568433

[30] Leslie Lamport. 2006. Fast Paxos. *Distributed Computing* 19, 2 (Oct. 2006), 79–103. https://doi.org/10.1007/s00446-006-0005-x

[31] Leslie Lamport, Dahlia Malkhi, and Lidong Zhou. 2009. Vertical Paxos and Primary-backup Replication. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing* (Calgary, AB, Canada) *(PODC '09).* ACM, New York, NY, USA, 312–313. https://doi.org/10.1145/1582716.1582783

[32] Barbara Liskov and James Cowling. 2012. *Viewstamped replication revisited.* Technical Report. Massachusetts Institute of Technology, Cambridge, MA, USA. http://pmg.csail.mit.edu/papers/vr-revisited.pdf

[33] Yanhong A. Liu, Saksham Chand, and Scott D. Stoller. 2019. Moderately Complex Paxos Made Simple: High-Level Executable Specification of Distributed Algorithms. In *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages 2019* (Porto, Portugal) *(PPDP '19).* ACM, New York, NY, USA, Article 15, 15 pages. https://doi.org/10.1145/3354166.3354180

[34] Yanhua Mao, Flavio P. Junqueira, and Keith Marzullo. 2008. Mencius: Building Efficient Replicated State Machines for WANs. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (San Diego, California) *(OSDI'08).* USENIX Association, Berkeley, CA, USA, 369–384. http://dl.acm.org/citation.cfm?id=1855741.1855767

[35] André Medeiros. 2012. *ZooKeeper's atomic broadcast protocol: Theory and practice.* Technical Report. Aalto University School of Science, Espoo, Finland. http://www.tcs.hut.fi/Studies/T-79.5001/reports/2012-deSouzaMedeiros.pdf

[36] Iulian Moraru, David G. Andersen, and Michael Kaminsky. 2013. There is More Consensus in Egalitarian Parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (Farminton, Pennsylvania) *(SOSP '13).* ACM, New York, NY, USA, 358–372. https://doi.org/10.1145/2517349.2517350

[37] Brian M. Oki and Barbara H. Liskov. 1988. Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing* (Toronto, Ontario, Canada) *(PODC '88).* ACM, New York, NY, USA, 8–17. https://doi.org/10.1145/62546.62549

[38] Diego Ongaro and John Ousterhout. 2014. In Search of an Understandable Consensus Algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (Philadelphia, PA) *(USENIX ATC'14).* USENIX Association, Berkeley, CA, USA, 305–320. https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro

[39] Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. 2017. Paxos Made EPR: Decidable Reasoning About Distributed Protocols. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 108 (Oct. 2017), 31 pages. https://doi.org/10.1145/3140568

[40] Robbert Van Renesse and Deniz Altinbuken. 2015. Paxos Made Moderately Complex. *ACM Comput. Surv.* 47, 3, Article 42 (Feb. 2015), 36 pages. https://doi.org/10.1145/2673577

[41] James R. Wilcox, Ilya Sergey, and Zachary Tatlock. 2017. Programming Language Abstractions for Modularly Verified Distributed Systems. In *2nd Summit on Advances in Programming Languages (SNAPL 2017).* Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 19:1–19:12. https://doi.org/10.4230/LIPIcs.SNAPL.2017.19