Brief Announcement: On the Significance of Consecutive Ballots in Paxos

Eli Goldweber
edgoldwe@umich.edu
University of Michigan
Ann Arbor, Michigan

Nuda Zhang
nudzhang@umich.edu
University of Michigan
Ann Arbor, Michigan

Manos Kapritsos
manosk@umich.edu
University of Michigan
Ann Arbor, Michigan

ABSTRACT

In this paper, we examine the Paxos protocol and demonstrate how the discrete numbering of ballots can be leveraged to weaken the conditions for learning. Specifically, we define the notion of consecutive ballots and use this to define Consecutive Quorums. Consecutive Quorums weaken the learning criterion such that a learner does not need matching *accept* messages sent in the *same ballot* from a majority of acceptors to learn a value. We prove that this modification preserves the original safety and liveness guarantees of Paxos. We define *Consecutive Paxos* which encapsulates the properties of discrete consecutive ballots. To establish the correctness of these results, in addition to a paper proof, we formally verify the correctness of a State Machine Replication Library built on top of an optimized version of Multi-Paxos modified to reflect *Consecutive Paxos*.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

Paxos, weakening, consensus, state machine replication

ACM Reference Format:
Eli Goldweber, Nuda Zhang, and Manos Kapritsos. 2020. Brief Announcement: On the Significance of Consecutive Ballots in Paxos. In *ACM Symposium on Principles of Distributed Computing (PODC '20), August 3–7, 2020, Virtual Event, Italy*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3382734.3405700

1 INTRODUCTION

More than 20 years after its inception, the Paxos algorithm [10, 13] remains a fundamental building block for distributed consensus and State Machine Replication (SMR) in an asynchronous setting. The importance of Paxos is made evident by the numerous variants of the algorithm — e.g. [8, 9, 12, 14] — and its use in real-world deployments [1–3]. Despite two decades of research on this topic, we have yet to understand all the subtleties of the Paxos algorithm.

The correctness of Paxos can be expressed concisely as "no two different values can be learned". This simple property, in turn, relies crucially on the way that values are learned:

- A value can only be learned if a majority of acceptors accepts that value *in the same ballot.*

The above criterion for learning is considered fundamental to the correctness of Paxos. And yet, this paper demonstrates that it is, in fact, *stronger* than it needs to be. Before we formally state and prove our claim, we will illustrate how this criterion is stronger than necessary by observing specific examples of Paxos in action.

Let us consider a number of Paxos snapshots, as seen from the perspective of a learner. Each snapshot shows the state of the five acceptors in a Paxos ensemble with $f = 2$. For each acceptor, we show the accepted value and the ballot in which this value was accepted. Since the learner only needs to receive an *accept* message from a majority of acceptors, we use "?" to denote the state of acceptors for which the learner has not received an *accept* message. For each snapshot, we consider the following question: *is it safe to learn value x?*

Acceptor ID	Value	Ballot
A	?	?
B	?	?
C	x	10
D	x	9
E	x	7

Table 1: Information collected at the learner. According to Paxos, value x cannot be learned yet. This is correct, since another value can still be learned.

Table 1 illustrates an example where the learner knows that a majority of acceptors have accepted the same value, but not in the same ballot. One might be tempted to conclude that, since a majority has already accepted x, it is henceforth impossible for any other value to be learned—and thus it is safe to learn x. But this not the case. Consider the following execution; initially, no acceptor has accepted a value. The proposer with ballot number 7 (henceforth, proposer #7) proposes x, which is accepted by E. Then proposer #8, who happens not to hear from E during phase one, is elected and proposes value y, which is accepted by acceptor B. At this point, proposer #9 performs phase one of Paxos and receives promise messages from acceptors C (-,-), D(-,-), and $E(x,7)$, and thus proposes value x, which is accepted by D. Similarly, proposer #10 receives promise messages from C(-,-), $D(x,9)$, and $E(x,7)$, and proposes value x, which is accepted by C. At this point, it is *not* safe to learn x, since it is still possible for a subsequent proposer, say proposer #11, to receive promise messages from A(-,-), $B(y,8)$, and $E(x,7)$, and to thus propose y.

PODC '20, August 3–7, 2020, Virtual Event, Italy

ACM ISBN 978-1-4503-7582-5/20/08…$15.00
https://doi.org/10.1145/3382734.3405700

Acceptor ID	Value	Ballot
A	?	?
B	?	?
C	x	10
D	x	9
E	x	9

Table 2: Information collected at the learner. According to Paxos, value x cannot be learned yet. And yet, *no other value* can be learned.

The above reasoning seems to lend credence to the criterion used in Paxos to determine whether a value can be learned: "a majority of acceptors must accept a value *in the same ballot*". A majority of acceptors that accept the same value in different ballots is not enough to merit learning that value. But what if the ballots are all *consecutive*?

Table 2 shows just such an example, where a majority of acceptors have accepted the same value, x, in consecutive ballots. In this paper we claim that in this case, x *can* be learned. The intuition behind this claim comes from observing the example of Table 1. The reason why x cannot yet be learned in that example is that there exists a ballot (#8 in this case) with a different value y, whose ballot number supersedes one of the ballots in the majority that has accepted x. This makes it possible that a future proposer picks this value y as its proposed value. When the majority consists only of consecutive ballots, however, no such "interleaved" ballot exists, which makes a *consecutive quorum* just as strong as if all ballots had the same ballot number.

We therefore propose the following weakening of Paxos — and all its corresponding variants. Given a Paxos ensemble with $2f + 1$ acceptors, a learner can learn a value x as soon as it receives $f + 1$ *accept* messages from distinct acceptors if: (a) all such messages denote acceptance of value x, and (b) the set of the ballot numbers of these messages consists of consecutive numbers. A more detailed version of this paper is available in [5].

2 MODEL AND OVERVIEW

We consider the model of Classic Paxos [13]. A system consisting of *proposer*, *acceptor*, and *learner* agents that communicate by passing messages over an asynchronous network. In typical implementations, agents are mapped to processes, where each process consists of one proposer, acceptor and learner agent.

In the Paxos protocol, proposers associate each proposed value v with a ballot number n, and acceptors accept proposals that are ballot-value pairs. We denote such a proposal as (n, v). It is important for different proposals to have different ballots, which can be achieved by having each proposer use ballots from its own disjoint pool.

In Classic Paxos, a value v is *chosen* if there exists a majority quorum of acceptors that have accepted the same proposal (n, v), that is, the same value v in the same ballot n. Once a value is chosen, two things occur. First, any pending proposals from a lower ballot will be ignored by at least a majority of acceptors. Any previous proposal can no longer achieve majority acceptance. Second, any higher numbered proposal must have a Phase 1 quorum that intersects with the quorum of acceptors that accepted the chosen proposal. This can only result in the proposer proposing the same v in Phase 2. Along with the requirement that any ballot be associated with a single value, once a value is chosen, no different value can ever be chosen or learned.

Given $2f + 1$ acceptors, Paxos is safe given the absence of malicious failures, and live given no more than f acceptors fail by crashing, during sufficiently long periods of synchrony [4].

The $Safety$ of Paxos is the property *"no two different values can be learned"*. A value is *learned* after a learner receives a quorum of *accept* messages indicating that a value has been chosen.

Theorem 2.1 (Safety Property). *No two different values can be learned*

$$\forall i, j \in \mathbb{N}.\ [Learned(i, v) \land Learned(j, w)] \implies v = w$$

To maintain the Safety Property, Paxos ensures *"no two different values can be chosen."* A learner does not learn a value until a value is chosen. If this invariant holds, then it must be the case that *"no two different values can be learned"*. Using this definition, the following is a formal description of the Chosen Invariant:

Theorem 2.2 (Chosen Invariant). *No two different values can be chosen*

$$\forall i, j \in \mathbb{N}.\ [Chosen(i, v) \land Chosen(j, w)] \implies v = w$$

In Classic Paxos, the following is the criterion that designates if a value is *chosen*:

Definition 1. [Chosen] Value v is *chosen* in ballot i iff a majority of acceptors send matching *accept* messages.

By proving the Chosen Invariant 2.2, the Safety Property 2.1 follows directly.

3 CONSECUTIVE QUORUMS

Consecutive ballots can be leveraged to weaken the criterion of how values are chosen and learned in Paxos. In this section we describe how Consecutive Quorums provide the same safety guarantee as majority quorums in Classic Paxos. By ensuring that no two different values can be chosen, at most one unique value can be learned. It is not necessary that a quorum of acceptors accept a value *in the same ballot* for the value to be learned. A value can be chosen—and eventually learned—as long as a majority of acceptors accept that value in an unbroken span of *consecutive* ballots. See [5] for a full proof of this.

In principle, the set of all ballots used in Paxos can be any unbounded, ordered set, including innumerable ones. However, it is ubiquitous in both theory [8, 11] and practice [1–3] to use as ballots the set of natural numbers, or any likewise discrete set. The additional structure that the natural numbers provide can be used to weaken the criterion for learning values in Paxos:

Definition 2. [Consecutive Quorum (CQ)] A quorum of acceptors $Q = \{\alpha_1, \ldots, \alpha_n\}$ is considered a *Consecutive Quorum* supporting the value v iff the set of ballots from the sent *accept* messages form a consecutive multiset.

Using this definition, we can weaken the criteria for a value to be considered *Chosen* in ballot i to:

Definition 3. [CQ Chosen] *A value is chosen in ballot i* once there is a majority quorum of acceptors that have sent *accept* messages with matching values and the ballots form a consecutive multiset. Additionally, at least one *accept* message must be sent in ballot i.

Note that *CQ Chosen* is strictly weaker than the Classic Paxos definition of *Chosen*. A classic majority quorum of matching *accept* messages is just a specific case of a Consecutive Quorum where all the ballots are of the same number.

The safety of Classic Paxos relies on the intersection of quorums. As long as a single quorum of acceptors sends matching *accept* messages to a learner, the chosen value is set in stone. Any future proposer, before proposing any value, will first obtain a quorum of *promise* messages. At least one acceptor will participate in both quorums. This ensures that only the chosen value could be proposed in a future ballot. A learner learns that value v is *CQ Chosen* in ballot i once receiving *accept* messages from a majority of acceptors, with matching values and the ballots form a consecutive multiset.

Learning with Consecutive Quorums does not affect the safety of Paxos. Consecutive Quorums are still majority quorums that will intersect with all other quorums. However, the ballot associated with the acceptor in the intersection might be different than in Classic Paxos without Consecutive Quorums. A valid Consecutive Quorum contains acceptors that have accepted the same value but from potentially different ballots. If the intersecting acceptor did not have the highest numbered proposal from the Consecutive Quorum, the definition of consecutive ballots ensures that there cannot exist any ballots between the reported ballot and the highest ballot in the Consecutive Quorum that could have a different value. As a result, *CQ Chosen* guarantees that once a value is chosen, no different value could also be chosen or learned.

Consider a set of 5 acceptors. For Classic Paxos to consider value v as chosen in ballot i, at least 3 acceptors must send *accept* messages for v in ballot i. In the case where the learner observes *accept* messages from distinct acceptors for value v in ballots $i-2$, $i-1$, and i, Classic Paxos cannot learn a value. However, This constitutes a valid Consecutive Quorum, and v would be considered learned. No other value could possibly be learned at this point.

4 ADDITIONAL USES OF CONSECUTIVE BALLOTS

Beyond weakening the criterion for learning values, consecutive ballots can be leveraged to weaken the criterion for proposing values. A proposer can immediately propose a value after receiving a *promise* message from the consecutive previous ballot. In fact, Heidi Howard in her thesis [7] observed that, once a proposer in ballot n receives a promise message from some ballot $m < n$, that any promise messages from ballots $< m$ contain no new information. In the case that ballots are discrete, this can be extended such that if m and n are consecutive, the proposer can safely proceed to Phase 2 immediately. This can take place even before the proposer has a full quorum of promise messages, which is the original criterion in Paxos. A proposer in Classic Paxos can proceed to Phase 2 when the classic criterion for proposing a value is met **or** the proposer receives a *promise* message from a consecutive previous ballot.

In addition to what Howard proposed, we show how the definition of Consecutive Proposals can be extended [5]. As observed in Fast Paxos [12] for reconfiguration, receiving an *accept* message relays the information that a *promise* message from the same acceptor would also contain. With our model, an agent can operate as multiple roles and could receive both *promise* and *accept* messages. This can be used to extend the definition of a Consecutive Proposal to be valid if the proposer observes a *promise* message from an acceptor that accepted a value in the previous ballot or an *accept* message from an acceptor in the previous ballot.

5 FORMAL VERIFICATION

Consecutive Quorums and Consecutive Proposals are not mutually exclusive. We define *Consecutive Paxos* as a protocol which incorporates both Consecutive Quorums and Consecutive Proposals.

To validate our confidence in Consecutive Paxos, we make use of formal verification to produce a mechanically-checked proof of its correctness. We build on top of the existing work from IronFleet [6], which showed that implementations of complex distributed systems can be formally verified. We prove that a replicated state machine built on an optimized version Multi-Paxos based on Consecutive Paxos maintains linearizability of client requests.

6 ACKNOWLEDGEMENTS

We thank the reviewers for their valuable feedback. We also thank Yixuan Chen for his critical insight on the example of Table 1, which served as inspiration for this work. This project was funded in part by the National Science Foundation under award CSR-1814507.

REFERENCES

[1] Jason Baker, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. 2011. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In *Proceedings of the Conference on Innovative Data system Research (CIDR)*. 223–234. http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf

[2] Mike Burrows. 2006. The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*. 335–350.

[3] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. 2013. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)* 31, 3 (2013), 1–22.

[4] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.

[5] Eli Goldweber, Nuda Zhang, and Manos Kapritsos. 2020. On the Significance of Consecutive Ballots in Paxos. *arXiv preprint arXiv:2006.01885* (2020).

[6] 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*. 1–17.

[7] Heidi Howard. 2019. *Distributed consensus revised.* Ph.D. Dissertation. University of Cambridge.

[8] Heidi Howard, Dahlia Malkhi, and Alexander Spiegelman. 2016. Flexible paxos: Quorum intersection revisited. *arXiv preprint arXiv:1608.06696* (2016).

[9] Heidi Howard and Richard Mortier. 2019. A Generalised Solution to Distributed Consensus. *arXiv preprint arXiv:1902.06776* (2019).

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

[11] Leslie Lamport. 2001. Paxos Made Simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)* (December 2001), 51–58. https://www.microsoft.com/en-us/research/publication/paxos-made-simple/

[12] Leslie Lamport. 2006. Fast paxos. *Distributed Computing* 19, 2 (2006), 79–103.

[13] Leslie Lamport et al. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.

[14] 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*. 358–372.