Classic Paxos vs. Fast Paxos: *Caveat Emptor*

Flavio Junqueira
Yahoo! Research Barcelona
fpj@yahoo-inc.com

Yanhua Mao
UC San Diego
maoyanhua@cs.ucsd.edu

Keith Marzullo
UC San Diego
marzullo@cs.ucsd.edu

Abstract

Classic Paxos and Fast Paxos are two protocols that are the core of efficient implementations of replicated state machines. In runs with no failures and no conflicts, Fast Paxos requires fewer communication steps for learners to learn of a request compared to Classic Paxos. However, there are realistic scenarios in which Classic Paxos has a significant probability of having a lower latency. This paper discusses one such scenario with an analytical comparison of the protocols and simulation results.

1 Introduction

With the advent of web services, grid services, and service-oriented architectures, fault-tolerant distributed services are now being deployed in many different network configurations. Consensus is the central protocol behind services replicated for fault tolerance. Hence, it is worth understanding how consensus performs across different network configurations. In this brief paper, we consider one point in this space: the latency of two versions of Paxos in one wide-area network configuration.

There are different versions of Paxos that have been designed for different environments and failure models. All provide safe, but not necessarily live, consensus in an asynchronous system. One, which is now usually called *Classic Paxos*, is relatively simple and has low latency in the normal case of timely message delivery and no failures. Another version, called *Fast Paxos*, has theoretically a lower message latency than Classic Paxos. It has some disadvantages over Classic Paxos, including needing a larger number of replicas, and having *collisions*, a scenario that can not occur in Classic Paxos and that leads to a higher message latency. Collisions in Fast Paxos are more likely to occur when the request rate is high.

It is not obvious, however, that the reduced message latency of Fast Paxos is in fact real when used in a wide-area network. Wide-area networks have a larger variance in message delivery than local-area networks. Depending on how the servers are distributed across the wide area and how the clients are distributed with respect to the servers, Fast Paxos may in fact have a higher message latency in practice. In this paper, we show why this is the case. This paper is meant to encourage further research in the area of the performance of core protocols across different network configurations.

2 Overview of the protocols

In this section, we give a quick overview of Classic Paxos [4] and Fast Paxos [5]. We only consider the failure-free case and concentrate on the message flow, the quorum size, and the replication requirements for both protocols.

Both Classic Paxos and Fast Paxos are designed for asynchronous consensus, where a group of processes propose values and eventually agree on a single proposed value. One of the most important applications of consensus is state-machine replication, where clients propose commands and the servers run a sequence of consensus instances. Each instance then selects a single command for the replicated service to execute.

Servers in both Classic Paxos and Fast Paxos have roles: there are *proposers* that propose values, *acceptors* that accept proposals and *learners* that learn the outcome of the consensus. Servers can take on all three roles. In Classic Paxos, a distinct proposer assumes the role of *leader*. Clients can send commands to all proposers, but only the leader actually proposes commands to acceptors. Accep-

	Classic Paxos	Fast Paxos
Comm. steps	3	2
Number of replicas	$2t+1$	$3t+1$
Quorum size	$t+1$	$2t+1$

Table 1. Summary of the Paxos protocols. The number of communication steps consider the case with no leader changes. Assuming that at most t processes can fail, the number of replicas corresponds to the minimum degree of replication to guarantee correctness, and the quorum size corresponds to the size of minimum subsets of acceptors necessary for progress.

tors can then accept the proposed value (they may not, for example, if there are multiple leaders). If a learner learns that a majority of acceptors has accepted the same value, it learns that the value is chosen and it knows that consensus is reached. In the steady state, it takes three communication steps ($client \rightarrow leader \rightarrow acceptor \rightarrow learner$) for a learner in Classic Paxos to learn the value.

Fast Paxos saves one communication step by allowing clients to propose values directly to the acceptors ($client \rightarrow acceptor \rightarrow learner$). However, to preserve safety, a larger quorum of acceptors is necessary. Assuming a threshold model that a maximum of t servers can fail, a learner needs to know that $2t+1$ acceptors have accepted the same value for it to know consensus has been reached, while in Classic Paxos the quorum size is $t+1$. This implies that Fast Paxos requires at least $3t+1$ acceptors while Classic Paxos requires only $2t+1$ acceptors. In addition, Fast Paxos can suffer from *collisions*, which can happen when two or more clients send proposals at nearly the same time, and acceptors receive these proposals in different orders. We do not discuss collisions in this short paper, even though the additional latency that arises from them can be large. Table 2 summarizes these facts on Classic and Fast Paxos.

3 Protocol analysis

Fast Paxos enables a learner to learn a new client request in two communication steps, whereas Classic Paxos requires three. In this section, we analyze the message latency of both flavors of Paxos, and we only discuss the case in which at most one server is faulty at any time. In this case, Classic Paxos requires a minimal of three servers while Fast Paxos requires four. We first introduce some notation. Let $lt(p_1, p_2)$ be the latency of a message sent from p_1 to p_2 and $pc(p_1)$ be the processing time of an operation in process p_1. We also use $smin\{A, B, C\}$ to denote the second smallest value among A, B, and C, and $tmin\{A, B, C, D\}$ to denote the third smallest value among A, B, C, and D.

We use the expression *learning latency* to denote the time for a given learner to learn a new request. For Classic Paxos, the learning latency is given by the following expression:

$$\begin{aligned} learn_{cp} &= lt(Client, Leader) + pc(Leader) \\ &+ smin\{A_1, A_2, A_3\} \end{aligned}$$

$$\begin{aligned} A_i &= lt(Leader, Acceptor_i) + pc(Acceptor_i) \\ &+ lt(Acceptor_i, Learner), i \in \{1, 2, 3\} \end{aligned}$$

If we assume that the processing times are negligible, then we can simplify the previous equation to the following:

$$learn_{cp} = lt(Client, Leader) + smin\{A_1, A_2, A_3\}$$

$$\begin{aligned} A_i &= lt(Leader, Acceptor_i) \\ &+ lt(Acceptor_i, Learner), i \in \{1, 2, 3\} \end{aligned}$$

For Fast Paxos, the equivalent expression is as follows:

$$learn_{fp} = tmin\{A_1, A_2, A_3, A_4\}$$

$$\begin{aligned} A_i &= lt(Client, Acceptor_i) + pc(Acceptor_i) \\ &+ lt(Acceptor_i, Learner), i \in \{1, 2, 3, 4\} \end{aligned}$$

If we assume that the processing times are negligible compared to message latencies, then we have the following:

$$learn_{fp} = tmin\{A_1, A_2, A_3, A_4\}$$

$$\begin{aligned} A_i &= lt(Client, Acceptor_i) \\ &+ lt(Acceptor_i, Learner), i \in \{1, 2, 3, 4\} \end{aligned}$$

To give an example in which Fast Paxos has a significant probability of having a higher latency compared to Classic Paxos, suppose that all servers are

in the same site, interconnected through a local-area network, and the client is in a different network, connected to the servers through a wide-area network. We can then assume that the network communication among the servers is negligible compared to the cost of the wide-area latencies. For Classic Paxos, we then have that the time for a learner to learn that a client request has been accepted is given by $learn_{cp} = lt(Client, Leader)$. For Fast Paxos, it is given by:

$$learn_{fp} = tmin\{lt(Client, Acceptor_i) : i \in \{1, 2, 3, 4\}\}$$

Suppose that wide-area latency ranges from α to β and follows a continuous probability density function (PDF) $D(x)$, where $x \in [\alpha, \beta]$. The cumulative distribution function (CDF) $C(x)$ of $D(x)$ captures the probability for a message to be delivered within time x, *i.e.*, $C(x) = \Pr(latency < x)$, where $x \in [\alpha, \beta]$.

The distribution of the learning latency for Classic Paxos is the same as the wide-area message latency distribution, as it is dominated by the message from the client to the leader and the communication among the servers is negligible. The PDF and CDF for Classic Paxos are then given by $D_{cp}(x) = D(x)$ and $C_{cp}(x) = C(x)$ respectively.

Assuming that the message latencies for Fast Paxos messages are independent, the CDF of the learning latency for Fast Paxos is as follows:

$$C_{fp}(x) = \Pr(learn_{fp} < x) = 4C^3(x)(1 - C(x)) + C^4(x)$$

Recall that the probability of Fast Paxos learning a value by time x is the probability of at least three out of four messages from the client to the servers being delivered within time x. The PDF is given by:

$$P_{fp}(x) = \frac{\mathrm{d}C_{fp}(x)}{\mathrm{d}x}$$

Now suppose an independent run of both protocols. The probability P that Fast Paxos is slower than Classic Paxos is:

$$P = \Pr(learn_{fp} > learn_{cp}) = \int_{x=\alpha}^{\beta} P_{fp}(x) C_{cp}(x) \mathrm{d}x$$

Noting that $C(\alpha) = 0$ and $C(\beta) = 1$, it is relatively simple to expand this equation and obtain:

$$P = 12 \int_{C(x)=0}^{1} (C^3(x) - C^4(x)) \mathrm{d}C(x) = \frac{3}{5}$$

This result implies that, independent of the distribution for wide-area message latencies, Classic Paxos is faster than Fast Paxos for 60% of the time in this particular network topology. Note that this proof only holds for continuous distributions, as if message latency is constant, for example, then the result clearly does not hold.

Intuitively, Fast Paxos has to wait for three messages out of four to make progress whereas Classic Paxos only requires one particular message. Even though the one message for Classic Paxos can be slow, this message is faster in most cases compared to waiting for three out of four messages.

An alternate way of obtaining this same result, but under different constraints is the following. Assume that the probability distribution of network latencies is not concentrated on any value. This assumption is weaker than assuming that the PDF is continuous, and it implies that if we sample the distribution multiple times, then all samples will be distinct with probability 1. Fast Paxos samples the network latency distribution four times. Let these samples have values A, B, C, D, where $A < B < C < D$. Classic Paxos samples the network latency distribution once. Let that sample have value E. By assumption, these five samples are distinct. There are five possible relations between the value E with respect to the other four values:

1. $E < A$;
2. $A < E < B$;
3. $B < E < C$;
4. $C < E < D$;
5. $D < E$.

Each case has the same probability because we draw A, B, C, D, E from the same distribution. Thus, the probability of Cases 1, 2 or 3 is $3/5$. These are the cases when Classic Paxos is faster than Fast Paxos.

4 Simulation

The performance of both variants of the Paxos algorithm depends upon how fast the network de-

livers messages to receivers. Their relative performance depends strongly on the variance of message latency. In many real networks, the variance in message latency is high due to traffic variations and non-deterministic scheduling of processes in a single computer. Informally, this observation implies that most of the time messages are delivered fast, but occasionally messages take one or two orders of magnitude more to be delivered.

In this section, we present simulation results on the latency of Classic Paxos and Fast Paxos. The simulator we use assumes that processing time is negligible compared to message latencies, and consequently the learning latency is the sum of the message latencies. To simulate message latencies, we considered traces obtained with NWS (Network Weather Service [8]) in the GrADS testbed [2] over the period between August and October of 2002. These are traces of TCP connections between pairs of machines. Each trace contains the time to establish a TCP connection, send four bytes, receive four bytes, and close the connection.

Our simulator is trace-driven. We associate the history between two computers in our dataset with a channel of our simulator, and for every message that crosses the channel, we obtain the latency for this message from the associated history. Also, we consider failure-free runs only, and we assume no conflicts between different clients. Failure-free runs should be the common case in many systems, and collisions introduce extra complexity into our environment not necessary to make our point. In fact, had we considered collisions for Fast Paxos, the latency for Fast Paxos would have been higher.

The case we present consists of a client and a set of servers implementing Paxos, where the client is in one site and all the servers are in another site. That is, only the communication between the client and the servers crosses a wide-area network. For this scenario, we have selected two different sites A and B from our dataset, and used the traces between two machines in different sites, one in A and one in B, and between pairs of machines in site B.

Figures 1 and 2 shows the cumulative fraction of requests (y-axis) with a given latency (x-axis). The latency of Figure 1 includes the time for a client to send a request to the servers implementing Paxos, the time for servers to exchange messages, and the time for a learner to learn this request by receiving accept messages from a quorum of acceptors. In addition to the latency mentioned for the case of Figure 1, the latency of Figure 2 also includes the time to send a response back to the client.

In these figures, if we draw a vertical line at some value of x_0, then the two y values of the two points in which this line crosses the curves correspond to the fraction of instances that Classic Paxos and Fast Paxos obtain a latency value $x \leq x_0$. From the learning curves, for values of $x_0 < 90ms$, the fraction of instances for which Classic Paxos obtain this latency is larger compared to the same fraction for Fast Paxos. The curves cross roughly at $90ms$ and, for values of $x_0 > 90ms$, the roles change, and the fraction of instances that have latency x or smaller is higher for Fast Paxos. The intuition for this result is as follows. Suppose we pick an instance of Classic Paxos as a reference, and consider the latency for a client request to reach the proposer. If the latency for this message is low, then there is a high probability that an instance of Fast Paxos using the same latency distribution is higher. This is due to the variance in message latency. As there are more messages from the client to the acceptors, the probability that at least two messages have a higher latency is significant compared to the request message to the proposer in Classic Paxos. If the request latency in Classic Paxos is high, then there is a high probability that Fast Paxos is faster because it can discard one message among all four sent to the acceptors if this message is too slow.

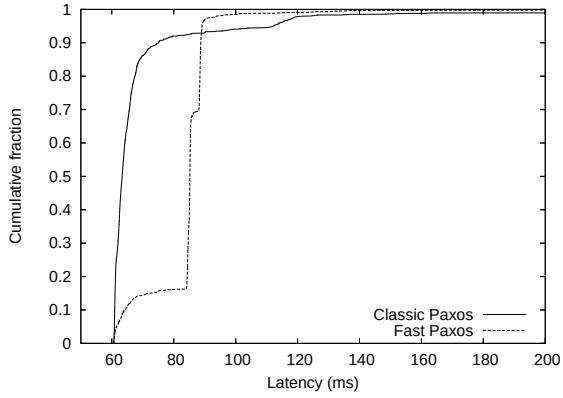

Figure 1. Cumulative distribution comparing Classic Paxos and Fast Paxos, learning latency.

Note also that the difference between Fast Paxos and Classic Paxos is more noticeable in the learning latency graph. For example, if we pick the value $x = 80ms$, the difference between the fraction of instances that have at most this learning latency value is over 0.7. In the client latency figure, the difference between y fraction values for the same latency value x is not greater than 0.5. This is due to addition of the latency to respond to the client in the client latency graph, which takes another wide-area communication step. This new step increases the variability in the latency of instances as now instances that are learned fast have a non-negligible probability of having slow wide-area messages on the way back to the client.

From a different perspective, we can also draw horizontal lines to determine the latency values for which we obtain a particular fraction of instances. For example, if $y = 0.5$, then in the learning latency graph Classic Paxos obtains this fraction with latency $62ms$, whereas Fast Paxos obtains this fraction with $83ms$. In general, for values of $y < 0.9$, the latency value for such a fraction is smaller for Classic Paxos in both graphs. Classic Paxos and Fast Paxos swap roles for $y > 0.9$.

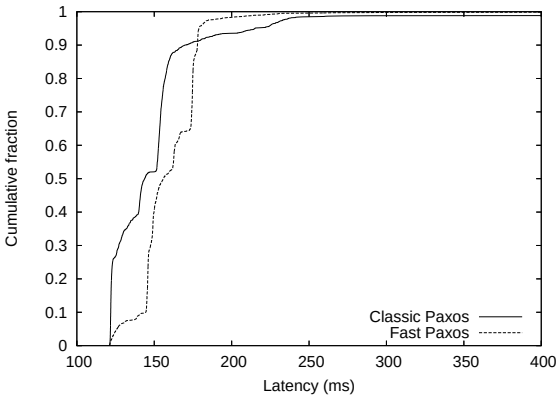

Figure 2. Cumulative distribution comparing Classic Paxos and Fast Paxos, client latency.

5 Future directions

Theoretically, counting the number of communication steps to learn the value is a simple way to evaluate the performance of consensus protocols. Indeed, in a system where system load is low and network delivery variance is small, communication steps can be an excellent predictor of latency. However, communication steps do not always translate directly into practical performance because of the asynchronous nature of the system. Especially in a wide-area network, there can be a large variance in message latency. In addition, often theoretical analysis overlooks other factors that can add to latency, such as available network bandwidth and operating system scheduling. In this section, we discuss a set of issues to consider when designing a consensus protocol for a given system.

Message complexity. Message complexity plays an important role in high-load systems. An increase in the number of messages by a constant factor can have significant performance penalty when system load increases. For example, Dobre *et al.* propose a new consensus algorithm that performs better under collisions [1], but requires a larger number of messages by a constant factor. Interestingly, their evaluation results show that this larger number of messages can cause a significant performance penalty when system load is high.

Quorum size. Larger quorum sizes require higher degrees of replication for the same degree of fault tolerance. In addition, in systems with significant variance in message delay and in operating system scheduling latency, a larger quorum size can lead to significantly higher latency [3].

Single critical path. The existence of single critical path makes performance vulnerable to a single performance problem along that path. Increased delay from the client to the leader in Classic Paxos slows the whole algorithm down. Classic Paxos can achieve 50% performance gain when it is able to dynamically change the leader [7].

Resistance to collision. Fast learning consensus protocols, such as Fast Paxos, rely on the absence of collisions to achieve high performance, and according to the Collision-Fast Learning Theorem by Lamport no general consensus algorithm can be fast learning upon collisions [6]. Collision is inherently hard to avoid in wide-area environments, in particular when there are multiple clients submitting requests concurrently. Because of large values and

high variance with message latency in wide-area environments, there is often a higher probability of collisions when using larger quorums such as the quorums that Fast Paxos requires.

Two possible approaches for achieving high performance across a wide range of environments are:

1. Run multiple protocols concurrently;
2. Switch between protocols depending on the network conditions.

For example, Fast Paxos outperforms Classic Paxos when both system load and message delay variance are low, but can do worse otherwise [1]. One can hope to achieve the better of two protocols at any time by using both protocols. However, this may not be ideal in practice. When running two protocols at the same time, additional care need to taken to make sure the two protocols choose consistent value. In addition, more messages will be sent when running the two at the same time, which may hurt the system performance as load increases.

Putting aside the policy of deciding when to switch, switching between protocols is straightforward when the consensus protocol is used to support replicated state machines. Switching can be learned as a proposal in the command sequence of the replicated state machine. A system is usually designed to have a maximum number (say α) of concurrent consensus instances, and so if a switching command is learned in instance i, switching can be achieved in instance $i + \alpha$. Furthermore, a *no-op* command can be used in bulk to fill the gap from instance $i + 1$ to $i + \alpha - 1$ so as to speed up the switching process. The less straightforward question is when to make the switch. We believe it is worth further study.

6 Conclusion

Classic Paxos and Fast Paxos are algorithms that enable efficient implementations of replicated state machines. Although intuitively Fast Paxos should always have a lower latency, this paper shows that increasing the number of messages from the client to the servers can result in a high number of instances of Classic Paxos having lower latency than Fast Paxos. This is due to the variability of message latency in networks. This variability increases the probability of high latency for an instance of Fast Paxos even in runs without collisions or failures. We have shown one wide-area scenario in which we can observe this effect. A more exhaustive study of scenarios in which this effect manifests, and practical corroboration of these results are subjects of future work.

Acknowledgements The alternate proof in Section 3 is due to Marcos Aguilera. We would like to thank him for his comments and also thank Russell Impagliazzo for the comments he gave us about this section. We also thank Rich Wolski for providing us with the NWS traces, and the reviewers for their helpful comments and good insight.

References

[1] D. Dobre, M. Majuntke, and N. Suri. CoReFP: Contention-resistant Fast Paxos for WANs. Technical Report TR-TUD-DEEDS-11-01-2006, Department of Computer Science, Technische Universität Darmstadt, 2006.

[2] F. B. *et al.* The GrADS Project: Software support for high-level grid application development. *International Journal of High Performance Computing Applications*, 15(4):327–344, Dec. 2001.

[3] F. Junqueira and K. Marzullo. Coterie availability in sites. In *Proceedings of DISC*, pages 2–16, Sept. 2005.

[4] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998.

[5] L. Lamport. Fast Paxos. *Distributed Computing*, 19(2):79–103, Oct. 2006.

[6] L. Lamport. Lower bounds for asynchronous consensus. *Distributed Computing*, 19(2):104–125, Oct. 2006.

[7] L. Sampaio and F. Brasileiro. Adaptive indulgent consensus. In *DSN'05*, pages 422–431, 2005.

[8] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *Proceedings of ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, Mar. 2003.