Summary on Extending Paxos to Tolerate Byzantine Faults

Paxos is generally used as a fault-tolerant solution to an asynchronous distributed system without Byzantine faults. However, it is possible to extend Paxos to tolerate Byzantine faults so that unpredictable Byzantine failures do not happen [1].

Recall that a Byzantine failure entails a faulty node that behaves unpredictably and misleads the rest of the nodes such that there is no solution to the consensus problem. Byzantine fault tolerance can be achieved with an algorithm by Miguel Castro and Barbara Liskov, where each node behaves like a replica of a state machine [2]. In this algorithm, a primary node multicasts each client request in three steps: *pre-prepare, prepare*, and *commit*. Each replica accepts the message, the *prepared* predicate becomes true, and it broadcasts a commit message. The system is successful when each replica has $2f+1$ correct messages (f being the number of faulty nodes), and the command is then executed.

To apply this algorithm to Paxos, an additional message delay is needed in the second step, *accept* [2]. Each acceptor needs to verify the value of each message before accepting it, so none of them gets tricked by a Byzantine node. To do so, each acceptor broadcasts the message to other acceptors and check if they received the same values. Consequently, when they send the *accepted* message to the proposer and learner, they would have found the discrepancy and chosen the correct value, so the Byzantine fault does not evolve into a failure.

Another interesting approach is to combine the Fast Paxos algorithm by Lamport [2] with the Fast Byzantine Consensus algorithm by Martin and Alvisi [3]. In this implementation, there is no longer a *proposer* role, and the initial *request* message and the *verify* message found in the previous algorithm are simplified away. In other words, there are three messages remaining: (1) a client sends a command to the acceptors, (2) the acceptors broadcast the values to each other and to the learner, and (3) the learner acts and sends the response to the client [4]. The diagrams below compare the normal (no Byzantine fault) flow of the two approaches, the left one using classical Paxos and the right one Fast Paxos (Figure 1). The speed advantage of using Fast Paxos is less distinct when a Byzantine fault exists, however.

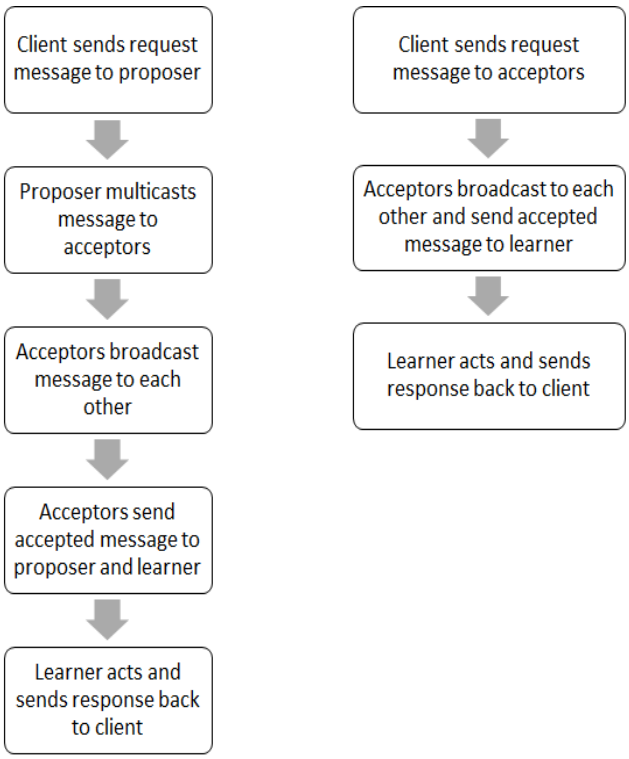

Figure 1: Comparison of classical Paxos and Fast Paxos without Byzantine faults

In conclusion, combining Paxos and Byzantine fault tolerance is a straightforward, although nontrivial, task. Although the combination could provide added security and reliability to a system, it would also add a significant message overhead. In turn, this would need to be mitigated by additional garbage collection and possibly hardware improvements. Additionally, the constraints of the original Byzantine General Agreement problem would still apply; the system would still fail when faulty nodes make up one third or more of the system.

REFERENCES

[1] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA*, 1999. Available: http://pmg.csail.mit.edu/papers/osdi99.pdf

[2] L. Lamport. Fast Paxos. In *Distributed Computing*, 2005. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2005-112.pdf

[3] Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, Yokohama, June 2005. IEEE Computer Society.

[4] Wikipedia contributors. Paxos (computer science). In *Wikipedia, The Free Encyclopedia*, 2017. Available: https://en.wikipedia.org/w/index.php?title=Paxos_(computer_science)&oldid=764100648