Leaderless Byzantine Paxos

Leslie Lamport

Microsoft Research

Appeared in *Distributed Computing: 25th International Symposium: DISC 2011*, David Peleg, editor. Springer-Verlag (2011) 141–142

Minor correction: 27 December 2011

Leaderless Byzantine Paxos

Leslie Lamport

Microsoft Research

Abstract. The role of leader in an asynchronous Byzantine agreement algorithm is played by a virtual leader that is implemented using a synchronous Byzantine agreement algorithm.

Agreement in a completely asynchronous distributed system is impossible in the presence of even a single fault [5]. Practical fault-tolerant "asynchronous" agreement algorithms assume some synchrony assumption to make progress, maintaining consistency even if that assumption is violated. Dependence on synchrony may be explicit [4], or may be built into reliance on a failure detector [2] or a leader-election algorithm. Algorithms that are based on leader election are called Paxos algorithms [6–8]. Byzantine Paxos algorithms are extensions of these algorithms to tolerate malicious failures [1, 9].

For Byzantine agreement, reliance on a leader is problematic. Existing algorithms have quite convincing proofs that a malicious leader cannot cause inconsistency. However, arguments that a malicious leader cannot prevent progress are not so satisfactory. Castro and Liskov [1] describe a method by which the system can detect lack of progress and choose a new leader. However, their method is rather ad hoc. It is not clear how well it will work in practice, where it can be very difficult to distinguish malicious behavior from transient communication errors.

The first Byzantine agreement algorithms, developed for process control applications, did not require a leader [10]. However, they assumed synchronous communication: that messages sent between nonfaulty processes are received within a known length of time. These algorithms are not suitable in the asynchronous case because a loss of synchrony can cause inconsistency.

We propose a simple method for implementing a leaderless Byzantine agreement algorithm: replacing the leaders in an ordinary Byzantine Paxos algorithm by a virtual leader that is implemented using a synchronous Byzantine agreement algorithm. Messages that in the ordinary algorithm are sent to the leader are instead sent to all the servers. Each server then decides what message the leader should send next and proposes it as the leader's next message. The servers then execute a synchronous Byzantine agreement algorithm to try to agree on the vector of proposed messages—a vector containing one proposal for each server. (This type of agreement is called *interactive consistency* [10].) Each server then uses a deterministic procedure to choose the message sent by the virtual leader, and it acts as if it had received this message.

When the system behaves synchronously, as is required for progress by any algorithm, each non-faulty server chooses the same virtual-leader message. The virtual leader thus behaves correctly, and the Byzantine Paxos algorithm makes progress. If the system does not behave synchronously, then the synchronous Byzantine agreement algorithm may fail, causing different servers to choose different virtual-leader messages. This is equivalent to a malicious leader sending conflicting messages to different processes. The malicious virtual leader can prevent progress (which cannot be guaranteed without synchrony), but does not cause inconsistency because a Byzantine Paxos algorithm can tolerate a malicious leader.

Leaderless Paxos adds to a Byzantine Paxos algorithm the cost of the leader agreement algorithm. The time required by a leader agreement algorithm that tolerates F faulty servers is $F+1$ message delays, which replaces the 1 message delay of a leader simply sending a message. (Early-stopping algorithms probably cannot be used because implementing a virtual leader seems to require simultaneous Byzantine agreement, which cannot guarantee early stopping [3].) For N servers, approximately NF extra messages are required.

References

1. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, pages 173–186. ACM, 1999.
2. Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
3. Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. Early stopping in Byzantine agreement. *Journal of the ACM*, 37(4):720–741, October 1990.
4. Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.
5. Michael J. Fischer, Nancy Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
6. Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998.
7. Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column)*, 32(4):51–58, December 2001.
8. Leslie Lamport. Fast paxos. *Distributed Computing*, 19(2):79–103, October 2006.
9. Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, pages 402–411, Yokohama, June 2006. IEEE Computer Society.
10. Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.