Brief Announcement: A Leader-free Byzantine Consensus Algorithm

Fatemeh Borran and André Schiper

Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
{fatemeh.borran,andre.schiper}@epfl.ch

We consider the consensus problem in a partially synchronous system with Byzantine faults. In a distributed system of n processes, where each process has an initial value, Byzantine consensus is the problem of agreeing on a common value, even though some of the processes may fail in arbitrary, even malicious, ways. It is shown in [11] that — in a synchronous system — $3t + 1$ processes are needed to solve the Byzantine consensus problem without signatures, where t is the maximum number of Byzantine processes. In an asynchronous system, Fischer, Lynch and Peterson [7] proved that no deterministic asynchronous consensus protocol can tolerate even a single non-Byzantine (= crash) failure. The problem can however be solved using randomization for benign and Byzantine faults. For Byzantine faults, Ben-Or [2] and Rabin [12] showed that this requires $5t + 1$ processes. Later, Bracha [3] increased the resiliency of the randomized algorithm to $3t + 1$.

In 1988, Dwork, Lynch and Stockmeyer [6], considered an asynchronous system that eventually becomes synchronous (called *partially synchronous system*). The consensus algorithms proposed in [6], ensure safety in all executions, while guaranteeing liveness only if there exists a period of synchrony. Recently, several papers have considered the partially synchronous system model for Byzantine consensus [4,10,8,1,5]. However, [1,5] point out a potential weakness of these Byzantine consensus algorithms, namely that they suffer from "performance failure". According to [1], a performance failure occurs when messages are sent slowly by a Byzantine leader, but without triggering protocol timeouts, and the paper points out that the PBFT leader-based algorithm [4] is vulnerable to such an attack. Interestingly, all deterministic Byzantine consensus algorithms for non-synchronous systems are leader-based. This raises the following fundamental question: is it possible to design a deterministic Byzantine consensus algorithm for a partially synchronous system that is not leader-based? With such an algorithm, performance failure of Byzantine processes might be harmless.

Results. Our results confirm the existence of a deterministic leader-free Byzantine consensus algorithm in a partially synchronous system that is resilient-optimal and signature-free. We started from the observation that leader-free consensus algorithms exist for the synchronous system, both for benign faults (e.g., the *FloodSet* algorithm [9]) and for Byzantine faults (e.g., the algorithm based on interactive consistency [11]). However, these algorithms violate agreement if executed during the asynchronous period of a partially synchronous

I. Keidar (Ed.): DISC 2009, LNCS 5805, pp. 479–480, 2009.

system. Therefore we tried to combine one of these algorithms with a second algorithm that ensures agreement in an asynchronous system.

We have applied our methodology by combining the synchronous consensus algorithm of [11] with a new algorithm that employs mechanisms from several consensus algorithms, e.g., Ben-Or [2], and PBFT [4] with strong validity. Let us denote these two algorithms by A_1, respt. A_2. Our combined algorithm is expressed in a round model. In each round, a correct process sends a message to all, receives a subset of messages sent, and computes its new state based on the messages received. Algorithm A_1 ensures that at the end of $t+1$ rounds, (i) if q is a correct process, any correct process p receives either v_q from q or nothing, where v_q is the value initially sent by process q, and (ii) if q is a faulty process, any correct process p receives either some common value v from q or nothing. Moreover, if all $t+1$ rounds are executed in synchronous periods, then all correct processes have the same set of messages at the end of $t+1$ rounds. Algorithm A_2 ensures safety (i.e., agreement and strong validity), while algorithm A_1 provides liveness (i.e., termination) during periods of synchrony. Our leader-free Byzantine consensus algorithm requires $3t+1$ processes, and $t+3$ rounds per consensus instance during periods of synchrony.

References

1. Amir, Y., Coan, B., Kirsch, J., Lane, J.: Byzantine Replication Under Attack. In: DSN 2008, pp. 197–206 (2008)
2. Ben-Or, M.: Another advantage of free choice (Extended Abstract): Completely asynchronous agreement protocols. In: PODC 1983, pp. 27–30. ACM, New York (1983)
3. Bracha, G.: An asynchronous [(n - 1)/3]-resilient consensus protocol. In: PODC 1984, pp. 154–162. ACM, New York (1984)
4. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance and Proactive Recovery. Transactions on Computer Systems (TOCS) 20(4), 398–461 (2002)
5. Clement, A., Wong, E., Alvisi, L., Dahlin, M., Marchetti, M.: Making Byzantine fault tolerant systems tolerate Byzantine faults. In: NSDI 2009, pp. 153–168. USENIX Association (2009)
6. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the Presence of Partial Synchrony. JACM 35(2), 288–323 (1988)
7. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of Distributed Consensus with one Faulty Process. J. ACM 32(2), 374–382 (1985)
8. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative byzantine fault tolerance. SIGOPS Oper. Syst. Rev. 41(6), 45–58 (2007)
9. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann, San Francisco (1996)
10. Martin, J.P., Alvisi, A.: Fast Byzantine Consensus. IEEE Transactions on Dependable and Secure Computing 3(3), 202–215 (2006)
11. Pease, M., Shostak, R., Lamport, L.: Reaching Agreement in the Presence of Faults. J. ACM 27(2), 228–234 (1980)
12. Rabin, M.: Randomized Byzantine generals. In: Proc. Symposium on Foundations of Computer Science, pp. 403–409 (1983)