Byzantine Replication for Trustworthy Systems

Jean-Philippe Martin, Lorenzo Alvisi
Laboratory of Advanced Systems Research
Department of Computer Sciences
University of Texas at Austin

1. Introduction

A trustworthy networked information system (NIS) must continue to operate correctly even in the presence of environmental disruptions, human errors, and hostile attacks [10]—in other words, it must be both fault-tolerant and secure. Reasoning about such a system is, unfortunately, very hard. Our limited success to date in building reliable NISs is an indication of the complexity involved with guaranteeing "just" fault-tolerance; once security is added, complexity runs the risk of becoming unmanageable. To compound the challenge, fault-tolerance and security do not seem to integrate nicely. On the one hand, they often duplicate efforts, since both are concerned with maintaining data integrity and availability. On the other hand, they can be at odds with each other—for instance, the very replication that improves data integrity and availability against faults, harms confidentiality.

An attractive approach toward managing this complexity is to model a component whose security has been compromised as faulty according to the Byzantine failure model. A system can then be hardened to guarantee its correct operation even if a subset of its components, up to a given threshold, suffer Byzantine faults.

This approach can potentially yield significant advantages. First, it holds the promise of simplifying reasoning about trustworthy systems by making security a byproduct of fault tolerance. Second, it opens the possibility of leveraging for security purposes the large body of existing research in Byzantine fault tolerance (BFT). Third, it suggests that it may be possible to assemble untrustworthy components into a trustworthy system, just as traditional fault-tolerance protocols assemble unreliable components to build reliable systems. This is especially attractive given the economics of today's marketplace, where few components are rigorously tested or verified.

At the same time, this approach raises several concerns. First, Byzantine fault tolerance is notoriously expensive. It is not just that Byzantine protocols, to operate correctly, typically require participants to engage in numerous rounds of communication; much more importantly, they require a very high degree of replication, The cost associated with this replication may be prohibitively high, especially when considering that *n*-version programming (or opportunistic n-version programming) may be required to reduce the possibility of a large number correlated Byzantine faults caused by a single security exploit.

Second, traditional Byzantine fault-tolerance techniques do not address confidentiality, which is a crucial aspect of security. Indeed, the replication required by most existing Byzantine fault-tolerance techniques can actually *hurt* confidentiality.

Third, the very soundness of modeling nodes compromised as a result of a security attack as if they were Byzantine faults is problematic. The ability to choose an accurate value for f, the maximum number of faults that can occur at any time, is critical to the safety of any BFT protocol. When Byzantine fault tolerance is not used against security attacks, it is reasonable to treat Byzantine failures as independent and compute f accordingly. But it is not obvious how to compute f so the system is safe when an attacker finds a new vulnerability in an operating system—when that happens, the probability that all replicas running the same OS will shortly be compromised increases sharply. It is not clear whether the diversity introduced through *n*-version programming is sufficient to reduce significantly the probability of this type of correlated faults.

Our research agenda over the last three years has been to determine whether or not hardening distributed systems against Byzantine faults is a practical and sound approach towards increasing trustworthiness. We have focused on two architectural primitives, state-machines and quorum systems, with encouraging results [1, 3, 2, 8, 5, 6, 11].

We are currently moving our agenda forward along three directions.

Stronger confidentiality We have shown [11] how, by restructuring state machine replication around separate agreement and execution replica clusters (see Figure 1), it becomes possible to insert between them a *privacy firewall*, whose mission is to filter responses so that Byzantine execution replicas cannot communicate confidential values to faulty clients. Our current firewall is

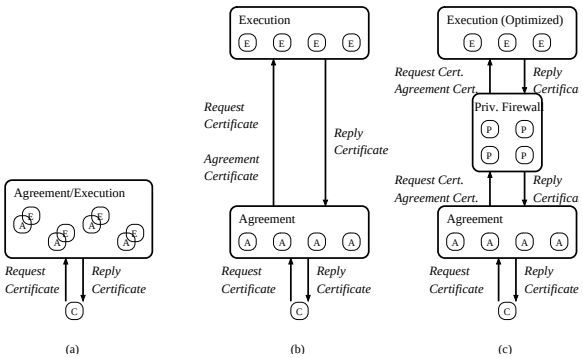

Figure 1: High level architecture of (a) traditional Byzantine fault tolerant state machine replication, (b) separate Byzantine fault tolerant agreement and execution, and (c) new optimizations enabled by the separation of agreement and execution.

still vulnerable to timing attacks and provides a level of confidentiality roughly equivalent to *possibilistic non-interference* [7].

We are investigating how to strengthen our firewall so that it becomes impossible for an adversary to affect the timing, ordering, and replication of symbols produced by our state machine. We propose to design and build an architecture that in a synchronous system guarantees *fail-safe confidentiality*—intuitively, when the correct component of the system meet their timeliness requirements, the system should be fully confidential; when they don't, the system should fall back to our version of possibilistic non-interference.

Better performance We have shown how the principle of separating agreement from execution can reduce significantly the replication costs incurred by services implemented as BFT state machines. Fundamentally, this separation allows our architecture to assign the tasks that require a high degree of replication—i.e. achieving the Byzantine agreement necessary to produce a linearizable order of client requests—to a cluster of generic *agreement nodes* that can be simple and inexpensive because they are application independent. *Execution nodes*, which sit behind the agreement cluster, are instead application-dependent and expensive, but fewer of them are necessary in our architecture than in earlier ones.

We are developing novel approaches to reduce the latency and increase the throughput of BFT state machines where, once again, the principle of separating agreement from execution enables promising directions for investigation.

A firmer foundation for BFT security The soundness of using BFT to build trustworthy NISs rests on a crucial assumption, namely, that failures across replicas are not highly correlated. It is indeed possible, as some have argued [9], that *n*-version programming combined with proactive recovery [4] provides sufficient diversity to inhoculate the NIS against correlated faults in the face of security exploits. However, to our knowledge, there is no quantitative evidence supporting this claim.

One of our goals is to determine the extent to which this claim is indeed defensible. To this end, we are currently working on developing a plausible methodology for quantifying the effectiveness of *n*-version programming against security exploits.

2. Fast Byzantine Paxos

As part of the agenda outlined above, we have recently developed FaB Paxos, a fast Byzantine Paxos protocol that, in the common case, is optimal in the number of communication steps required to reach consensus. Since consensus is at the core of state machine replication, FaB Paxos leads to state machine replication protocols that have lower common case latency than the current ones.

To properly discuss FaB Paxos, a quick review of Paxos is in order. Paxos [2] phrases the consensus problem in terms of the actions taken by three classes of agents: *proposers*, who propose values, *acceptors*, who together are responsible for choosing a single proposed value, and *learners*, who must learn the chosen value. Informally, the protocol's safety properties require that only a single value be chosen from among those that are proposed and that only chosen values be learned; liveness instead consists in making sure that eventually, proposed values will be chosen and chosen values will be

learned. More specifically, in Paxos one of the proposers is elected leader and it communicates with the acceptors. Paxos guarantees progress only when the leader is unique and can communicate with sufficiently many acceptors, but it ensures safety even with no leader or with multiple leaders.

One of the properties that make Paxos attractive for practical applications is that it can be made very efficient in *gracious executions*, i.e. executions where there is a unique correct leader, all correct acceptors agree on its identity, and all correct replicas and all links between them are timely.

Except in pathological situations, it is reasonable to expect that gracious executions will be the norm—and so it is desirable to optimize for them. Indeed, it has been shown that for in non Byzantine environment, Paxos-like consensus protocol can terminate successfully after only two communication steps (matching the lower bound for the problem) in the common case of gracious executions.

Unfortunately, this is not the case for Byzantine versions of Paxos, like the ones at the core of PBFT [1] and of the state machine replication protocol that we have proposed in a recent paper [3]. After a client request has been received by the leader, Byzantine protocols require a minimum of three additional communication steps before the reply can be determined.

The difficulty in making Byzantine protocols fast in the common case is that they must continue to be correct in the uncommon case. For instance, while in traditional Paxos a faulty leader can at most be silent, in Byzantine Paxos one has to consider scenarios in which a faulty leader may (i) try to communicate inconsistent values to the acceptors, possibly poisoning their state forever; (ii) try to trick acceptors into assigning the same sequence number to two distinct requests; or (iii) collude with faulty acceptors to have distinct correct learners learn different values. Byzantine protocols use the extra round precisely to defend against these and other similar eventualities.

Our approach for breaking this stalemate is unconventional: we use a higher number of acceptors nodes than in traditional protocols and treat the resulting set of acceptors as a Byzantine quorum system. FaB Paxos uses the extra acceptors as a sort of "institutional memory" for the actions taken by the leader. If the leader is faulty, correct nodes of good can query appropriately this quorum system to obtain an accurate account of the status of the system, allowing us to eliminate in the process the need for an extra communication step.

For traditional implementations of the state machine approach, the extra replication required by our approach would probably make the tradeoff uninteresting, except perhaps for the applications most bent on reducing latency. However, in Byzantine state machines where agreement and execution are separated, as is the case for the architecture that we have recently proposed [3], acceptors correspond to the cheap agreement nodes and can be used more liberally—making the design point that we explore much more attractive.

References

[1] L. Alvisi, D. Malkhi, E. Pierce, and M. Reiter. Probabilistic techniques for fault detection in Byzantyne quorum systems. In *Proceedings of the Seventh IFIP International Working Conference on Dependable Computing for Critical Applications (DCCA-7)*, pages 357–372, January 1999.

[2] L. Alvisi, D. Malkhi, E. Pierce, and M. Reiter. Fault detection for Byzantine quorum systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(9):996–1007, September 2001.

[3] L. Alvisi, D. Malkhi, E. Pierce, M. Reiter, and R. Wright. Dynamic Byzantyne quorum systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000 and FTCS 30)*, pages 283–292, June 2000.

[4] Y. Huang, C. Kintala, N. Kolettis, and N.D. Fulton. Software rejuvenation: analysis, module, and applications. In *Proceedings of the 25rd Fault-Tolerant Computing Symposium*, pages 381–390, June 1995.

[5] J.P. Martin, L. Alvisi, and M. Dahlin. Small Byzantyne quorum systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2002 and FTCS 32), DCC Track*, pages 374–383, June 2002.

[6] J.P. Martin, L. Alvisi, and M. Dahlin. Small Byzantyne quorum systems. In *Proceedings of the 16th International Symposium on Distributed Computing (DISC 2002)*, pages 311–326, October 2002.

[7] J. McLean. A general theory of composition for a class of 'possibilistic' security properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, January 1996.

[8] E. Pierce and L. Alvisi. A framework for semantyc reasoning about Byzantyne quorum systems. In *Brief Announcements, Proceedings of the 20th ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 317–319, August 2001.

[9] R. Rodrigues, M. Castro, and B. Liskov. BASE: Using abstraction to improve fault tolerance. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, 2001 October.

[10] Fred B. Schneider, editor. *Trust in Cyberspace*. National Academy Press, 1999.

[11] J. Yin, J.P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating agreement from execution for Byzantine fault-tolerant services. In *Proceedings of the Nineteenth Symposium on Operating System Principles*, pages 253–268. ACM SIGOPS, October 2003.