From Byzantine-Tolerant to Intrusion-Safe Services

Christian Cachin

IBM Research - Zurich
cca@zurich.ibm.com

18 September 2009

Byzantine-fault-tolerant services. Byzantine fault-tolerance (BFT) protocols let a set of n redundant replicas act together for implementing a service accessed by clients, so that the service tolerates that up to f replicas fail. Failures may be crashes, value-domain errors, the result of attacks, or even intrusions by a malicious adversary.

Many protocols and practical systems for tolerating Byzantine faults have been developed recently; most of them avoid timing assumptions (they work in an asynchronous model) and employ cryptographic authentication. Typically, such protocols tolerate only less than $n/3$ faulty replicas.

Assumptions. The typical f-out-of-n failure assumption is justified when replicas fail independently of each other, in other words, when the likelihood that a replica fails is the same for all replicas, regardless of which failures have already occurred. This assumption is well justified for crash failures and holds often in practice. But when failures are the consequences of malicious attacks, then care has to be taken to ensure that replicas respond independently to attacks. It means typically to place the n replicas in n different administrative domains, with different machine configurations and/or operating systems. Otherwise, the cost of successfully attacking *all* replicas is not much higher than the cost of the first intrusion [2, 8].

But what if more than a fraction of the replicas fail? For machines connected to the Internet, it seems almost impossible that a system runs over a long time and no more than a third of the replicas *ever* fail. Periodic system rejuvenation with proactive recovery [5, 6, 4] has been proposed to reduce the duration of this vulnerability from the system lifetime to a shorter time period. However, this essentially re-introduces a timing assumption into an area that has taken many efforts to eliminate timeouts! The question of semantics beyond f failures remains and is left wide open by the state-of-the-art BFT systems.

Fail-aware untrusted services. In recent work addressing non-replicated service providers, Cachin, Keidar, and Shraer [3] have introduced the notion of a *fail-aware untrusted service* that gives meaningful semantics even when the provider is faulty. In the common case, when the provider is correct, such a service guarantees consistency (i.e., linearizability) and liveness (i.e., wait-freedom) of all operations. Should the provider be faulty, however, the clients eventually detect any inconsistency and the service makes the clients aware of the protocol violation by the provider.

The same paper presents a fail-aware untrusted storage service (FAUST), which relies on so-called forking semantics and provides eventual consistency and failure awareness. FAUST can be extended in a simple way, by storing the sequence of all client requests to the service, to implement any fail-aware untrusted service described by a state machine, although the resulting protocol is not efficient.

Intrusion-safe services. Combining the properties of BFT systems with fail-aware untrusted services, we conceive a class of systems that we call *intrusion-safe*. They implement a service using n replicas, of which up to f may fail in arbitrary ways. An intrusion-safe service tolerates up to f replica failures, maintaining liveness and consistency (i.e., linearizability) in the same way as traditional BFT systems. When more than f replicas fail, the service still guarantees a degraded notion of consistency (formulated using forking semantics), but not necessarily liveness. As for a fail-aware untrusted service, the operations are eventually consistent and the clients can eventually detect any protocol violation.

Intrusion-safe services relax the "all-or-nothing" semantics of BFT systems under attack and guarantee a precise notion of graceful degradation. This property is important for high-resilience systems in practice and has been the focus of recent research [9, 1]. Specifically, when f or more — even up to n — replicas are faulty, an intrusion-safe service enables the clients to eventually detect any inconsistency and alerts the clients that the fault-tolerance assumption ("up to f faults") has been violated. Previous work on BFT protocols could only guarantee a variant of forking semantics with less than $2n/3$ faulty replicas [7].

We have developed an efficient *weak fork-linearizable untrusted service protocol*, based on the principles from the efficient untrusted storage protocol (USTOR), which underlies FAUST [3]. It serves as our basis to implement an intrusion-safe service in a modular way: take any BFT atomic broadcast protocol, let the replicated servers run the weak fork-linearizable untrusted service protocol on top of it, and enhance the clients with a fail-awareness layer.

Acknowledgments. This presentation is based on joint work with Idit Keidar and Alex Shraer (Department of Electrical Engineering, Technion).

Part of this work was done at and supported by the Distributed Programming Laboratory (LPD), School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL).

References

[1] A. S. Aiyer, E. Anderson, X. Li, M. A. Shah, and J. J. Wylie, "Consistability: Describing usually consistent systems," in *Proc. 4th Workshop on Hot Topics in Syetms Dependability (HotDep)*, 2008.

[2] C. Cachin, "Distributing trust on the Internet," in *Proc. International Conference on Dependable Systems and Networks (DSN-DCCS)*, pp. 183–192, 2001.

[3] C. Cachin, I. Keidar, and A. Shraer, "Fail-aware untrusted storage," in *Proc. International Conference on Dependable Systems and Networks (DSN-DCCS)*, 2009.

[4] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous verifiable secret sharing and proactive cryptosystems," in *Proc. 9th ACM Conference on Computer and Communications Security (CCS)*, pp. 88–97, 2002.

[5] R. Canetti, R. Gennaro, A. Herzberg, and D. Naor, "Proactive security: Long-term protection against break-ins," *RSA Laboratories' CryptoBytes*, vol. 3, no. 1, 1997.

[6] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, pp. 398–461, Nov. 2002.

[7] J. Li and D. Mazières, "Beyond one-third faulty replicas in Byzantine fault-tolerant systems," in *Proc. 4th Symp. Networked Systems Design and Implementation (NSDI)*, 2007.

[8] F. B. Schneider and L. Zhou, "Implementing trustworthy services using replicated state machines," *IEEE Security & Privacy Magazine*, pp. 34–43, Sept. 2005.

[9] L. Zhou, V. Prabhakaran, V. Ramasubramanian, R. Levin, and C. A. Thekkath, "Graceful degradation via versions: Specifications and implementations," in *Proc. 26th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 264–273, 2007.