Byzantine Fault-Tolerance with Commutative Commands

Pavel Raykov[1], Nicolas Schiper[2], and Fernando Pedone[2]

[1] Swiss Federal Institute of Technology (ETH)
Zurich, Switzerland
[2] University of Lugano (USI)
Lugano, Switzerland

Abstract. State machine replication is a popular approach to increasing the availability of computer services. While it has been largely studied in the presence of crash-stop failures and malicious failures, all existing state machine replication protocols that provide byzantine fault-tolerance implement some variant of atomic broadcast. In this context, this paper makes two contributions. First, it presents the first byzantine fault-tolerant generic broadcast protocol. Generic broadcast is more general than atomic broadcast, in that it allows applications to deliver commutative commands out of order—delivering a command out of order can be done in fewer communication steps than delivering a command in the same order. Second, the paper presents an efficient state machine replication protocol that tolerates byzantine failures. Our protocol requires fewer message delays than the best existing solutions under similar conditions. Moreover, processing of commutative commands on replicas requires only two MAC operations. The protocol is speculative in that it may rollback non-commutative commands.

1 Introduction

State machine replication is a popular approach to increasing the availability of computer services [1,2]. By replicating a service on multiple machines, hardware and software failures can be tolerated. Although state machine replication has been largely studied in the presence of crash-stop failures and malicious failures, all existing protocols that provide byzantine fault-tolerance (BFT) (e.g., [3,4,5,6,7]) implement some variant of atomic broadcast, a group communication primitive that guarantees agreement on the set of commands delivered and on their order. In this context, this paper makes two contributions.

The first contribution of this paper is a byzantine fault-tolerant generic broadcast protocol. Generic broadcast defines a conflict relation on messages, or commands, and only orders messages that conflict. Two messages conflict if their associated commands do not commute. For instance, two increment operations of some variable x commute since the final value of x is independent of the execution order of these operations. Generic broadcast generalizes atomic broadcast—the two problems are equivalent when every two messages conflict. Previous generic

A. Fernández Anta, G. Lipari, and M. Roy (Eds.): OPODIS 2011, LNCS 7109, pp. 329–342, 2011.

broadcast protocols appeared in the crash-stop model [8,9,10]; ours is the first to tolerate malicious failures. The difficulty with generic broadcast stems from the need to deliver commutative commands in two communication delays and ensure that their delivery order, with respect to non-commutative commands, is the same at all correct processes. To address this challenge under byzantine failures we define Recovery Consensus, an abstraction that ensures proper ordering between conflicting and non-conflicting messages. The proposed protocol requires $n \geq 5f + 1$ replicas to tolerate f byzantine failures. We use Recovery Consensus at the core of our generic broadcast protocol.

The second contribution of this paper is a state machine replication protocol that generalizes and improves current byzantine fault-tolerant state machine replication protocols. Our protocol builds on our generic broadcast algorithm. A naive implementation of state machine replication based on generic broadcast to propagate commands to servers would lead to a best latency of three communication delays. We rely on speculative execution to provide an efficient algorithm that executes commutative commands in two communication delays. The algorithm is speculative in that it may rollback commands in some cases (i.e., when non-commutative commands are issued). To summarize, the principal advantage of the proposed state machine replication protocol is to allow fast execution of commutative commands in two message delays. Moreover, when commands commute servers only need to execute two MAC operations per command.

The remainder of the paper is structured as follows. Section 2 defines the system model. Sections 3 and 4, respectively, present the Recovery Consensus and generic broadcast protocols. We extend our generic broadcast protocol to provide state machine replication in Section 5. Section 6 discusses related work and Section 7 concludes the paper. Correctness proofs of the protocols can be found in the appendix of the full version of this paper [11].

2 System Model and Definitions

We consider an asynchronous message passing system composed of n processes $\Pi = \{p_1, \ldots, p_n\}$, out of which f are *byzantine* (i.e., they can behave arbitrarily). A process that is not byzantine is *correct*. The adversary that controls byzantine processes is computationally bounded (i.e., it cannot break cryptographic primitives) and cannot change the content of messages sent by one correct process to another correct process. The network is fully connected and *quasi-reliable*: if a correct process p sends a message m to a correct process q, then q receives m.[1] We make use of public-key signatures to allow a process to sign a message m [12]. We denote message m signed by process p_i as $\langle m \rangle_{\sigma_i}$. We also use HMACs [13] to establish a bidirectional authenticated channel between any two processes p_x and p_y, with the notation $\langle m \rangle_{\sigma_{xy}}$ indicating a message m signed with a secret key shared between processes p_x and p_y.

[1] The presented algorithms can trivially be modified to tolerate *fair-lossy* links, links that may drop messages but guarantee delivery of a message m if m is repeatedly sent. We assume quasi-reliable links to simplify the presentation of the algorithms.