Brief Announcement: Replacement - Handling Failures in a Replicated State Machine

Leander Jehl, Tormod Erevik Lea, and Hein Meling

University of Stavanger, Norway
{leander.jehl,tormod.e.lea,hein.meling}@uis.no

1 Introduction

State machine replication is a common approach for building fault-tolerant services. A Replicated State Machine (RSM) typically uses a consensus protocol such as Paxos [1] to decide on the order of updates and thus keep replicas consistent. Using Paxos, the RSM can continue to process new requests, as long as *more than half of the replicas* remain operational. If this bound is violated, however, the current RSM is forced to stop making progress indefinitely. To avoid scenarios in which the number of failures exceeds the bound, it is beneficial to immediately instantiate failure handling, if this can be done without causing a significant disruption to request execution.

This can be done by reconfiguration, which is a general method to replace one set of replicas with another. Classical reconfiguration relies on the RSM to decide on a reconfiguration command [2]. For this, the old configuration must have a majority of operational replicas and a single correct leader. The latter can only be guaranteed if the replicas are sufficiently synchronized.

In this paper, we present Replacement [3], a reconfiguration algorithm specialized for replacing a faulty replica with a new one. Also Replacement requires a majority of operational replicas. However, different from traditional reconfiguration techniques, failure handling with Replacement does not rely on consensus. Thus, by using Replacement, faulty replicas can be replaced even during times of asynchrony, e.g. when clocks are not synchronized and the network experiences unpredictable delays, or when multiple replicas are competing for leadership. This is useful, since replacing slow or overloaded replicas can restore synchrony and replaced replicas can no longer compete for leadership.

In [4] we showed that reconfiguration without consensus is possible. However, the algorithm presented in [4] (ARec), has to stop the state machine during reconfiguration. Replacement, our new method, includes minor adjustments to the Paxos algorithm that allow the RSM to make progress, while replicas disagree on the current configuration. It thus avoids the increased client latency and temporary unavailability, caused by ARec.

2 Contribution

Replacement is similar to the round change in Paxos. A replacement request, specifying an old replica and its replacements, is propagated to all replicas, which

then send PROMISE messages to the new replica. The new replica can determine a correct state and start running Paxos, after collecting a quorum of promises. The following ideas are key to Replacement.

New state only for the new replica. To ensure that no different values can get chosen before and after the replacement, we guarantee that a value, accepted by a majority before replacement, is still accepted by a majority after replacement. For this, it is enough if the new replica stores any possibly accepted value. Therefore, in Replacement, only the new replica needs to wait for promises, while the other replicas can continue to run Paxos.

Vector Timestamps. In Replacement, replicas use a vector clock to timestamp the current configuration. By attaching this vector clock to messages, we can detect and discard messages from replaced replicas. Thus Replacement can allow replicas, that are not replaced, to continue running Paxos in the same round. This is different from other reconfiguration methods [4,5] which enforce a round change in Paxos, and thus discard all messages from the previous round.

Combining Replacements. Every replacement has a unique timestamp and if two concurrent replacements are issued for the same replica, the one with the higher timestamp will be executed. However, if two concurrent replacements are issued for different replicas, both replacements will be executed, possibly in different orders. Thus, replacements for different replicas can be issued by different agents, without the risk that some replacement is lost due to concurrency with another, unrelated replacement.

Since replacements for different replicas are executed concurrently without any order or priority, concurrent replacements can block each other. We solve this with simple coordination among the replacing processes, which is only necessary if a majority of the replicas are replaced concurrently.

Evaluation Our evaluation shows that using ARec causes longer repair times and temporary unavailability, compared to classical reconfiguration. Replacement performs on par with classical reconfiguration in a synchronous setting, but also allows failure handling in times of asynchrony.

References

1. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. **16**(2) (May 1998) 133–169
2. Lamport, L., Malkhi, D., Zhou, L.: Reconfiguring a state machine. SIGACT News **41**(1) (March 2010) 63–73
3. Jehl, L., Meling, H.: Towards fast and efficient failure handling for paxos state machines. In: Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on. (2013) 98–102
4. Jehl, L., Meling, H.: Asynchronous Reconfiguration for Paxos State Machines. In: ICDCN. (2014) 119–133
5. Lamport, L., Malkhi, D., Zhou, L.: Vertical paxos and primary-backup replication. In: PODC. (2009) 312–313