Partition tolerance and execution semantics for state-machine replication

Franz J. Hauck [1], Jannis Dommer[1], and Alexander Heß [1]

Abstract: There are a couple of serious frameworks supporting the use of state-machine replication (SMR) in order to build fault-tolerant services. In theory, the underlying consensus protocols allow for partitions on one hand. On the other hand, most frameworks claim to achieve exactly-once execution semantics. Unfortunately, both is not correctly implemented most of the time. Partition tolerance is most likely a very desirable property. However, it is debatable whether exactly-once semantics is actually necessary and achievable, especially when it comes to long lasting network partitions. The proposed presentation will sched some light on three frameworks with regard to both aspects: BFT-SMaRt, PBFT in its TinyBFT version, and Themis.

State-machine replication is a well-known concept to build fault-tolerant services [Sc90]. As the replicated state machines, for short the replicas, may be disconnected by network partitions, it is an interesting question how the system behaves when partitions occur and last for some significant amount of time. These frameworks typically use a totally-ordered multicast protocol based on consensus algorithms to achieve deterministic input to the replicated state machines. In the theory of these consensus algorithms partitions induce no problem as quorums are necessary to achieve progress. If the algorithms cannot collect these quorums they get stuck until the quorum is reachable. In practice, however, some frameworks rely on TCP, but forget to resend messages after a TCP connections is broken. This is especially unfortunate when partitions are over.

On the other hand, SMR systems are services that receive requests and return responses. This raises the question about execution semantics as it has been discussed for RPC-based systems. For a fault-tolerant system, exactly-once semantics seems to be possible as there is always a couple of replicas active and not faulty. However, with the appearance of partitions this assumption is not necessarily true. Thus, it is an interesting question how existing SMR implementations handle this aspect.

In order to get insights into the corresponding capabilities of current SMR frameworks, we have looked at BFT-SMaRt [BSA14] in Versions v1.2 and v2.0, PBFT [CL02] as it was updated to modern build chains in the TinyBFT project [BDW24], and Themis [RMK18]. It turns out that BFT-SMaRt and Themis have serious problems with partitions and cannot fullfil their promises about execution semantics. Even worse, during our analyses for this

1 Ulm University, Institute of Distributed Systems, Germany,
franz.hauck@uni-ulm.de, https://orcid.org/0000-0002-7480-9617; jannis.dommer@outlook.de;
alexander.hess@uni-ulm.de, https://orcid.org/0000-0001-6837-2861

work, we could identify serious bugs that either compromise the assumptions about the failure model or lead to deadlocks and fatal aborts that prevent the application from making progress.

Our presentation will address our findings, uncover serious problems of two of the three mentioned frameworks and discuss possible solutions.

Bibliography

[BDW24] Böhm, Harald; Distler, Tobias; Wägemann, Peter: TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems. In: 30th IEEE Real-Time & Emb. Techn. & Appl. Symp. (RTAS). 2024.

[BSA14] Bessani, Alysson Neves; Sousa, João; Alchieri, Eduardo Adílio Pelinson: State Machine Replication for the Masses with BFT-SMaRt. In: 44th Ann. IEEE/IFIP Int. Conf. on Dep. Syst. & Netw. (DSN). pp. 355–362, 2014.

[CL02] Castro, Miguel; Liskov, Barbara: Practical Byzantine Fault Tolerance and Proactive Recovery. ACM Trans. Comp. Syst., 20(4):398–461, nov 2002.

[RMK18] Rüsch, Signe; Messadi, Ines; Kapitza, Rüdiger: Towards low-latency Byzantine agreement protocols using RDMA. In: 48th Ann. IEEE/IFIP Int. Conf. on Dep. Syst. & Netw. Workshops (DSN-W). pp. 146–151, June 2018.

[Sc90] Schneider, Fred B.: Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Comp. Surv., 22(4):299–319, December 1990.