Poster: Recover from Excessive Faults in Partially-Synchronous BFT SMR

Tiantian Gong*, Kartik Nayak†, Gustavo Franco Camilo*, Andrew Lewis-Pye‡, Aniket Kate*§

*Purdue University {*tg, gfrancoc, aniket*}*@purdue.edu*

†Duke University *kartik@cs.duke.edu*, ‡London School of Economics *a.lewis7@lse.ac.uk*, §Supra Research

I. Problem statement

Byzantine-fault tolerant (BFT) state machine replication (SMR) [8], [2] protocols solve the problem of replicating the same state consistently among a distributed system of n nodes, called replicas, while tolerating up to f of them being Byzantine. Here, Byzantine replicas behave arbitrarily (including the possibility of crash faults). A secure BFT SMR protocol achieves safety, where replicas output consistent transaction logs, and liveness, where transactions input to a sufficient number of correct replicas are eventually output by correct replicas. SMR utilizes total order broadcast or atomic broadcast (ABC) or consensus primitives to order transactions submitted by clients. In an asynchronous network, consensus is impossible with a deterministic protocol, even in the presence of a single crash failure [5].

One approach [2], [10], [6] to circumventing this impossibility is to relax the network asynchrony assumption to partial synchrony, where there is a known bound on message delivery after some unknown global stabilization time (GST). But BFT SMR is still impossible for more than one-third of faults in partially-synchronous networks. Set $f = \lceil \frac{n}{3} \rceil - 1$ and let the actual number of Byzantine faults be f_a. When $f_a > f$, both safety and liveness violations can arise. Previous works have focused on safety violations and studied accountability for BFT protocols in this setting.

Accountability informally means that all correct replicas eventually identify a set of faulty parties with a proof of culpability, when correct replicas equivocate and commit to contradicting outputs. In setting $f_a \leq n-2$,[1] Civit et al. [4] propose a generic algorithm achieving accountability for general decision task protocols, including Byzantine agreement. This general approach incurs at least an $O(n^2)$ multiplicative communication complexity overhead, and triples the round complexity. Civit et al. [3] later improves on communication overhead by adding two extra rounds to the base protocol, after which an output is finally committed.

The open problem is then in setting $f_a \leq n - 2$, whether we can embed accountability in general non-synchronous BFT SMR protocols more efficiently without altering the committing rules in normal executions, and further, whether replicas can recover from equivocating states safely. We first study the *feasibility and complexity of fault detection in general partially-synchronous BFT SMR protocols* for $f_a \leq n - 2$. We then discuss achieving *safety and liveness under a stronger commit rule by recovering from safety violations* in quorum-based SMR protocols while considering excessive alive-but-corrupt (ABC) faults [7]. Here, ABC faults target at compromising safety but not liveness.

II. Approach to solve the problem

The first challenge is to achieve sound and effectively complete failure detection in setting $(f_a \geq \lceil \frac{2n}{3} \rceil)$. *Effective completeness* means identifying at least $(f+1)$ faulty replicas in case of safety violations among correct replicas. *Soundness* is satisfied if correct replicas are never identified as culpable. The difficulty of ensuring both effective completeness and soundness in this setting is that the faulty replicas can already write arbitrary history (except for $f < 2$) and lead correct replicas to send specific messages. Here, they can generate arbitrary history because a secure SMR protocol achieves liveness, and $(n - f)$ replicas can make progress. We tackle this challenge by preserving evidence of malicious behaviors among correct replicas, by letting correct replicas only accept a message with sufficient causal histories.

The second challenge is to recover from equivocations while achieving safety and preserving past progress. SMR protocols are not one-shot and replicas continuously process client requests and commit outputs. In case of equivocations, naïvely rolling back to the last agreed location and remove faulty replicas disregard

[1] Note that the problem is not well-defined for $f_a \geq n - 1$.

TABLE I
GENERIC APPROACHES FOR REALIZING ACCOUNTABILITY. (THE MESSAGE IN A PROTOCOL IS OF SIZE $O(z)$.)

Methods	# Faults	Communication overhead	Round overhead	Client aid
[9] Protocol-dependent algorithms that analyze existing messages	$2f$	-	-	Yes
[4] Reliable-broadcast each outgoing message piggybacked with newly received messages	$n-2$	$\times\ O(n^2)$	$\times\ 3$	No
[3] Add two extra confirmation rounds to any consensus protocol	$n-2$	$+\ O(n^2)$	$+\ 2$	No
This work: Indicate causal history in outgoing messages	$n-2$	Black-box: $\times\ O(zn)$ Non-black-box: $+\ O(n^3)$	-	No

past progress and does not ensure that replicas have the same view on which replicas are to blame. We repair equivocating logs while preserving past progress via a deterministic recovery algorithm. We adopt a weaker safety and liveness definition called "safety and liveness under strong commit" for $f_a > f$. Here, "strong commit" for a block informally means that sufficient number of replicas votes for it or its children but not contradicting sibling blocks. For liveness, we consider ABC faults that only intend to break safety.

The third challenge is to ensure small communication, computation, and storage overhead, which facilitates adoptions of protocols. Forming proofs of culpability or recovering from faults in a non-synchronous network requires the preservation and dissemination of previously received messages, and computation on those messages. To reduce overhead, we study in-place fault detection that utilizes the existing communication rounds, adopt simple detection and recovery routines, and devise garbage collection routines for recycling storage.

III. Preliminary results

First, we uncover sufficient conditions for a *general* BFT SMR protocol to allow for effectively complete and sound fault detection upon safety violations when $f_a \leq n-2$. This means blaming at least $(f+1)$ faulty parties when the equivocating replicas provide all pertinent data, and never blaming correct replicas. Second, we provide a deterministic recovery algorithm for fixing equivocations in *quorum-based* SMR while preserving past progress. It is built upon a fault detector and a strictly monotone branch selector which picks one of the contradicting history branches. By running the recovery algorithm locally, correct replicas eventually adopt the same transaction log and expels the same set of faults. We instantiate the theory in Tendermint [1] and HotStuff [10] by adding fault detection and recovery routines to the two protocols.

The fault detection and recovery algorithms can treat an SMR protocol as a black box. The fault detection algorithm can also utilize existing communication rounds in the original protocol, reducing its communication complexity overhead. The instantiated fault detection routines have $O(n^2)$ additive authenticator complexity and $O(\kappa n^3)$ additive bit complexity if the protocol terminates in $O(n)$ rounds. Our recovery algorithm carries over the complexity of the fault detection algorithm and has an extra $O(n)$ additive authenticator complexity for repairing equivocations. To reduce storage overhead, we provide garbage collection routines: in recovery, logs up to the last strongly committed block can be recycled.

References

[1] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[2] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[3] P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, and J. Komatovic, "As easy as abc: Optimal (a) ccountable (b) yzantine (c) onsensus is easy!" *Journal of Parallel and Distributed Computing*, vol. 181, p. 104743, 2023.

[4] P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic, Z. Milosevic, and A. Seredinschi, "Crime and punishment in distributed byzantine decision tasks," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 34–44.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[6] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 45–58.

[7] D. Malkhi, K. Nayak, and L. Ren, "Flexible byzantine fault tolerance," in *Proceedings of the 2019 ACM SIGSAC CCS*, 2019, pp. 1041–1053.

[8] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, 1990.

[9] P. Sheng, G. Wang, K. Nayak, S. Kannan, and P. Viswanath, "Bft protocol forensics," in *Proceedings of the 2021 ACM SIGSAC CCS*, 2021, pp. 1722–1743.

[10] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM PODC*, 2019, pp. 347–356.

✦Purdue University, ✳Duke University, ❀London School of Economics, ✤Supra Research

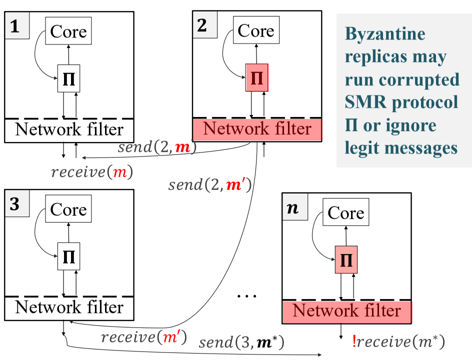
Recover from Excessive Faults in Partially-Synchronous BFT SMR

Tiantian Gong✦, Kartik Nayak✳, Gustavo Franco Camilo✦, Andrew Lewis-Pye❀, Aniket Kate✦,✤

OVERVIEW

We analyze BFT SMR in excessive fault setting, i.e., where the actual number of Byzantine faults surpasses a protocol's tolerance.

- We first establish ***sufficient conditions*** for a general BFT SMR protocol to allow for ***effectively complete*** *(blaming many Byzantine nodes)* ***and sound*** *(i.e., not blaming correct nodes)* ***fault detection*** when up to $(n-2)$ Byzantine replicas attack safety.
- We then relax the safety definition in SMR to ***safety under strong commit*** and devise a deterministic repair procedure for recovering from excessive faults.



Partially-Synchronous BFT SMR with Excessive Faults

▷ **Partial synchrony**: ∃ GST Δ and a known finite delay δ such that
after Δ, each message sent at time t^* is delivered to the receiver by $(t^*+\delta)$

▷ **Excessive faults**: out of n replicas, $f_a > \frac{n}{3}$ are Byzantine

Impossibility – In partial synchrony, SMR ***cannot*** be achieved (i.e., safety and liveness violations occur) in setting $f_a > n/3$

▷ **Accountability**: **Locate faulty parties** after safety violations

▷ **Recovery**: **Recover** from safety violations

Goal

(a) Effectively complete and **sound** fault detection for $f_a \le n-2$

- **Effective completeness** – locate $(f+1)$ faulty parties given data from **two** equivocating parties
- **Soundness** – **no** correct party mis-identified as Byzantine

(b) Recovery for $f_a \le \frac{2n}{3}$

(c) Efficiency in both fault detection and recovery

Solution overview

(a) General Fault Detection

A. Sufficient conditions for complete and sound fault detection in general SMR protocols

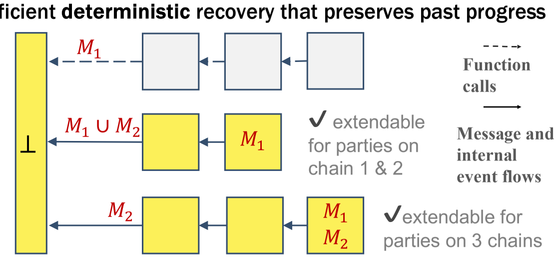
B. Both black-box and non-black-box (efficient) approaches

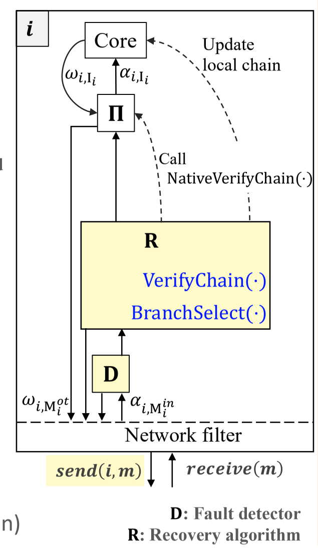
Methods	# Faults	Communication overhead	Round overhead	Client aid
[1] Protocol-dependent algorithms analyzing existing messages	$2f$	–	–	Yes
[2] Reliable broadcast each outgoing message piggybacked with newly received messages	$n-2$	$\times O(n^2)$	×3	No
[3] Add 2 extra confirmation rounds	$n-2$	$+O(n^2)$	2	No
This work: indicate immediate causal histories	$n-2$	Black-box: $\times O(zn)$ * Non-black-box: $+O(n^3)$	–	No

Table I. Generic approaches for realizing accountability. * z is the size of messages

(b) Recovery for $f_a \le \frac{2n}{3}$ in Quorum-Based BFT SMR

A. Efficient deterministic recovery that preserves past progress





B. Achieves safety and liveness under strong commit

i. **Strong commit**
A block with sufficient "endorsers" is strongly committed

ii. **Safety under strong commit**
Correct parties strong commit consistent logs

ii. **Liveness under strong commit**
Assuming ABC faults, correct parties make progress

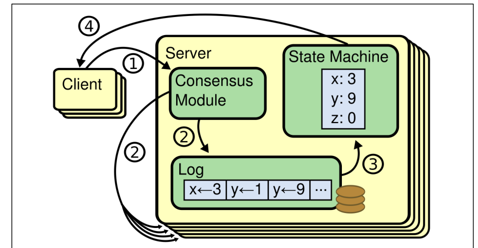
C. Instantiations in two SMR protocols
Tendermint (all-to-all) and HotStuff (linear communication)

State Machine Replication

SMR: Emulate a single always-correct machine with a known set of replicas that may become faulty, e.g., crash or corrupted by a Byzantine adversary.

Security of SMR

- **Safety** – correct parties commit consistent transaction logs
- **Liveness** – correct parties make progress in committing transactions



Raft [Ongaro-Ousterhout14]

Evaluation

We evaluated the performance of general fault detection routines in HotStuff on AWS m5d.8xlarge machines, running Ubuntu, 22.04 LTS, amd64 jammy image. We ran a total of 20 machines in North Virginia, North California, Ohio, Singapore, Sydney, Ireland, and London.

★ **Compared with benchmark, the throughput is not affected but the added routines induce a higher latency.**

References

[1] P. Sheng, G. Wang, K. Nayak, S. Kannan, and P. Viswanath, "Bft protocol forensics," in Proceedings of the 2021 ACM SIGSAC CCS, 2021, pp. 1722–1743.
[2] P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic, Z. Milosevic, and A. Seredinschi, "Crime and punishment in distributed byzantine decision tasks," in 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS). IEEE, 2022, pp. 34–44.
[3] P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, and J. Komatovic, "As easy as abc: Optimal (a) ccountable (b) yzantine (c) onsensus is easy!" Journal of Parallel and Distributed Computing, vol.181, p. 104743, 2023.