

Towards Recoverable Hybrid Byzantine Consensus

Hans P. Reiser
LaSIGE, University of Lisboa
hans@di.fc.ul.pt

Rüdiger Kapitza
Informatik 4
University of Erlangen-Nürnberg
rrkapitz@cs.fau.de

Abstract

In the recent years, a lot of research effort has been spent on finding better BFT consensus algorithms. Most approaches try to improve performance under specific conditions, but ignore the problem of how to recover from faults. In practice, replicas in a BFT system not only need to recover from detected faults (reactive recovery), but also from undetected faults (proactive recovery). For a wide deployment of BFT systems, a closer analysis of this problem is compulsory.

1. Problem statement

Malicious attacks on distributed computing systems are a serious problem. BFT state-machine replication is a well-understood approach for implementing intrusion-tolerant applications that operate correctly even if some nodes are faulty. Without a mechanism for recovering from malicious faults, sooner or later, an attacker may gain control over more replicas than the system can tolerate. Due to the inherent difficulty of accurately detecting intrusions, replicas should therefore *proactively* be refreshed periodically, besides *reactively* repairing detected faults.

Proactive recovery in BFT systems was first addressed by PBFT [1]. This work identifies a set of prerequisites that are needed for incorporating proactive recovery into the basic PBFT algorithm:

- a) A tamper-free device periodically triggers recoveries.
- b) A trusted component securely stores a private key; this component creates signatures on behalf of a replica, without revealing the key to a malicious intruder if the replica is faulty.
- c) An attacker, who controls a faulty replica and thus may sign messages using the trusted component, is prevented from using these messages later (after the recovery of the node) for impersonating that node.
- d) A persistent state storage permits a correct replica to keep its state even if it is recovered.
- e) Replicas must continue processing messages while they are recovering; this ensures that all replicas eventually

recover, even if more than f nodes simultaneously enter the recovering state.

All these prerequisites, except the first one, are tightly linked to the internals of a BFT algorithm. This observation implies that support for recovery is an integral feature of an algorithm, and not an orthogonal mechanism that can easily be developed independently.

2. Analysis of existing systems

Table 1 gives an overview of recovery support of BFT algorithms, starting with the original PBFT and including all major subsequent improvements. The *TCB* column indicates the use of a trusted component at each node.

Surprisingly, no recent improvement of PBFT addresses the problems of recovering faulty nodes. Q/U explicitly allows correct nodes to crash and recovery. This support, however, does not include means for a faulty node to get its state repaired and return to correct operation again. HQ, BFT2F, and Zyzzyva are improvements of PBFT using a lightweight quorum-based protocol when there is no contention, a weaker consistency model that allows remaining safe with up to $2f$ faults, and speculative execution, respectively. None of these three systems consider recoveries at all.

A2M [6] uses a TCB (attested append-only memory) that enables consensus with only $2f + 1$ nodes. Essential protocol state is stored within the TCB and cannot be modified by an attacker, which might simplify recoveries. The paper, however, does not discuss any details of repairing faulty nodes. MinBFT [7] achieves BFT replication with $2f + 1$ using a simpler TCB than A2M, but also does not support recoveries.

Algorithm	TCB	Recovery support
PBFT [1]	yes	yes
Q/U [2]	no	no
HQ [3]	no	no
BFT2F [4]	no	no
Zyzzyva [5]	no	no
A2M [6]	yes	maybe
MinBFT [7]	yes	no

Table 1. Recovery support in BFT algorithms

Observation 1: The successors of PBFT make great strides in improving some performance aspects, but fail to address the important question of recovering faulty nodes.

PBFT already identified the need for a tamper-free device that periodically triggers recoveries.

Observation 2: Recoverable BFT systems must have a hybrid architecture with a trusted component.

A2M and MinBFT have shown that a TCB allows reducing the number of required replicas from $3f + 1$ to $2f + 1$. In combination with Observation 2, it is reasonable to predict that practical recoverable BFT systems will use a hybrid architecture and require $2f + 1$ replicas.

3. Building recoverable BFT systems

Of the prerequisites listed in Section 1, only (a) is independent of the algorithm itself. Items (b-d) require internal modifications at the interface for key generation, the validity verification of signatures, and the way internal protocol state is stored, respectively. While it might be feasible to apply these changes to existing protocols in a mostly generic way, this is not true for the last item (e). Functionality for correcting potentially corrupt local state is not part of protocols not aware of recoveries. A corresponding protocol extension is non-trivial and requires detailed understanding of the protocol and its state.

Figure 1 shows a state model for recoverable BFT consensus algorithms. Faulty nodes can be repaired and return to correct operation (transition $D \Rightarrow C \Rightarrow B$). An additional challenge arises from the attempt to timely recover replicas in an asynchronous system model. Recoveries (transitions $B \Rightarrow A$, $A \Rightarrow A$, $D \Rightarrow C$ and $C \Rightarrow C$) can be triggered periodically using a synchronous watchdog device. On the other hand, the recovery execution (verifying and if needed repairing the node state in transitions $C \Rightarrow B$ and $A \Rightarrow B$) may take an unbounded time under the asynchronous model.

- The PBFT solution *assumes* that at most f nodes simultaneously enter state C or D . These nodes eventually finish their recoveries and return to state B .
- Assuming a synchronous network for recoveries is a strong modification of the system model, but would allow *completing* recovery operations timely.

Observation 3: Recoverable BFT algorithms must bridge the gap between the asynchronous operation of the network and the synchronous nature of proactive recoveries. Future BFT research needs to address this question in more detail.

4. Conclusions

The integrated support of proactive recovery is a prerequisite for a wide deployment of practical Byzantine replication systems. With the notable exception of the original

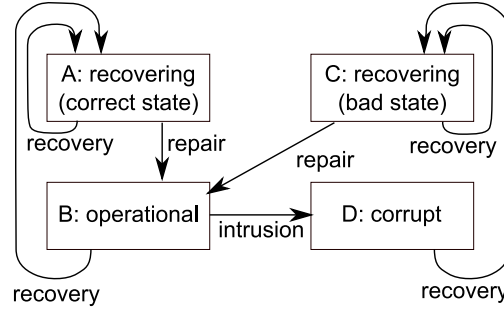


Figure 1. State transition model of a recoverable BFT algorithm

PBFT algorithm, currently existing BFT algorithms do not consider this functionality. This paper briefly sketched some challenges of recoverable BFT consensus algorithms. We observe that the support for recovery is an integral feature of an algorithm, not an orthogonal mechanism that can easily be developed independently. The inherent need for a TCB in recoverable algorithms also justifies using $2f + 1$ -algorithms in practical systems. As a side effect of reducing the number of replicas, it becomes more feasible to satisfy replica diversity, another prerequisite for practical deployment of BFT replication.

References

- [1] M. Castro and B. Liskov, “Practical Byzantine fault tolerance and proactive recovery,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
- [2] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, “Fault-scalable Byzantine fault-tolerant services,” *SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, pp. 59–74, 2005.
- [3] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira, “HQ replication: A hybrid quorum protocol for Byzantine fault tolerance,” in *Proc. of the 7th Symp. on Operating Systems Design and Impl. (OSDI)*, Seattle, Washington, Nov. 2006.
- [4] J. Li and D. Mazières, “Beyond one-third faulty replicas in Byzantine fault tolerant systems,” in *4th Symp. on Networked Systems Design and Impl. (NSDI)*, 2007.
- [5] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, “Zyzyva: speculative Byzantine fault tolerance,” *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 45–58, 2007.
- [6] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz, “Attested append-only memory: making adversaries stick to their word,” in *SOSP ’07: Proc. of 21st ACM SIGOPS Symp. on operating systems principles*. ACM, 2007.
- [7] G. S. Veronese, M. Correia, L. C. Lung, and A. N. Bessani, “Minimal Byzantine fault tolerance,” Department of Informatics, University of Lisbon, DI/FUL TR 08–29, Dec 2008.