Testing the “Fast Byzantine Consensus” Protocol

Author: Alexandra Carutasu <a.carutasu@student.tudelft.nl>
Supervisors: Dr. Burcu Kulahcioglu Özkan, João M. Louro Neto

Date:31.01.2025

1 Introduction

- The “Fast Byzantine Consensus” protocol (FaB) was introduced in 2006 by Martin et al. [1].
- It is the first protocol to achieve consensus in just two communication steps under Byzantine assumptions.
- Abraham et. al uncovered a “bounded liveness” violation in their theoretical analysis of the protocol [2]; no other bugs have been detected since then.

2 Research Objective

The aim of this research was to evaluate **ByzzFuzz**’s performance as a testing framework. We devised the following RQs:

1. Can ByzzFuzz find any bugs in the implementation of the given protocol?
2. How does the bug detection performance of ByzzFuzz compare to a baseline testing method that arbitrarily injects network and process faults?
3. How do small-scope and any-scope message mutations of ByzzFuzz compare in their performance of bug detection for the given protocol?

3 Methodology

- Implemented the “Fast Byzantine Consensus” protocol using **ByzzBench**, adapted the implementation to a multi-shot consensus protocol.
- Ran experiments using the two testing methods: **ByzzFuzz** and **baseline**, compared the performance of the two testing approaches.
- Tested the protocol using *small-scope* or *any-scope* mutations in **ByzzFuzz**.

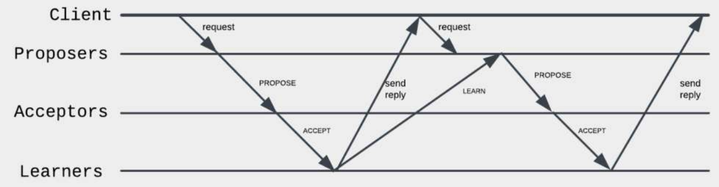

Figure 1. Common case execution of our implementation of the Fast Byzantine Consensus multi-shot adaptation

4 Results

- Focused on testing the non-parametrized version of FaB.
- Ran the experiments under **f = 1,** in a network with **6** nodes and faults injected among **r = 10** rounds of execution for **3000** scenarios.

1. Can ByzzFuzz find any bugs in our implementation of the FaB protocol?

- Yes, injecting process and network faults during the execution using **ByzzFuzz** uncovered both liveness and disagreement violations.

Figure 2. ByzzFuzz execution results (*p* and *n* indicate the number of process and network faults injected)

- We traced back the root cause of the violations to oversimplifications in our adaptation to the multi-shot consensus variant.
- No bugs in the original protocol design were found.

2. How does the bug detection performance of ByzzFuzz compare to a baseline testing method that arbitrarily injects network and process faults?

- **ByzzFuzz** was more effective, identifying both disagreement and liveness violations
- The **baseline** method failed to detect any new liveness or any disagreement violations.

Faults	# of violations detected	detection rate
p = 1, n = 0	20	0.67%
p = 2, n = 0	30	1.00%
p = 0, n = 1	139	4.63%
p = 0, n = 2	203	6.77%
p = 1, n = 1	133	4.75%
baseline	4	0.13%

Table 1. ByzzFuzz and arbitrary fault injection comparison

3. How do small-scope and any-scope message mutations of ByzzFuzz compare in their performance of bug detection for the FaB protocol?

- We reran **ByzzFuzz** for **1000** scenarios, alternating between *small-scope* and *any-scope* message mutations.
- Small-scope mutations performed better, uncovering the disagreement scenarios in our implementation

	process faults	liveness	disagreement	detection rate
small-scope	1	1	7	0.8%
	2	3	8	1.1%
any-scope	1	5	0	0.5%
	2	5	0	0.5%

Table 2. Small-scope and any-scope mutations comparison

- Using ByzzBench, we materialized the liveness violation uncovered by Abraham. et al [2].

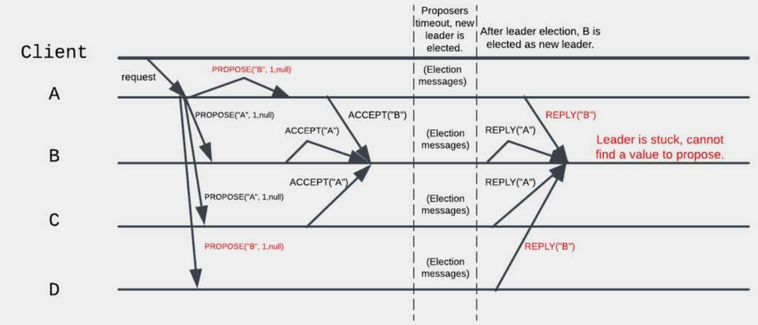

Figure 3. Materialzied bug in our implementation of the Fast Byzantine Consensus protocol

5 Conclusions

- ByzzFuzz uncovered both liveness and disagreement terminations in our implementation.
- ByzzFuzz was more efficient than the baseline testing method.
- Small-scope mutations were more efficient in uncovering disagreement violations in our implementation.

6 Limitations

- Our implementation for handling of multi-shot operations in the protocol is too simplistic, needs enhacement.
- Refine ByzzBench’s liveness detection capabilities.

[1] I. Abraham, G. Golan-Gueta, D. Malkhi, L. Alvisi, R. Kotla, and J.-P. Martin, “Revisiting fast practical byzantine fault tolerance,” ArXiv, vol. abs/1712.01367,2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:7902429
[2] J.-P. Martin and L. Alvisi, “Fast byzantine consensus,”IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 3, pp. 202–215, 2006