Evaluating the correctness and safety of hBFT with ByzzFuzz.

Attila Birke
A.B.Birke@student.tudelft.nl

Dr. Burcu Kulahcioglu Özkan
João Miguel Louro Neto

TU Delft

1. Background

- Distributed systems are used all around the world, in financial transactions, cloud computing, etc.
- Byzantine Fault Tolerance (BFT) allows a distributed system to withstand several Byzantine Faults.
- Testing is crucial to ensure the safety of BFT algorithms.
- Lack of automated testing algorithms.
- hBFT [1] is a leader-based protocol that uses speculation.
- ByzzFuzz [2] is a randomised testing algorithm, which uses round-based structure-aware small-scope mutations.

2. Research Questions

- **RQ1: To what extent is ByzzFuzz able to evaluate the correctness and safety of hBFT?**
- RQ2: Can ByzzFuzz find any bugs in the implementation of the hBFT protocol?
- RQ3: How does the bug detection performance of ByzzFuzz compare to a baseline testing method that arbitrarily injects network and process faults?
- RQ4: How do small-scope and any-scope message mutations of ByzzFuzz compare in their performance of bug detection for hBFT?

3. Method

- Implemented the hBFT protocol in ByzzBench.
- Implemented structure aware mutations.
- Tested hBFT with ByzzFuzz and baseline testing methods.
- Evaluated the difference between small-scope and any-scope message mutations.

Message	Mutations
<PREPARE, v, n, d(m), m, c>	<PREPARE, **v'**, n, d, m, c>
	<PREPARE, v, **n'**, d, m, c>
<COMMIT, v, n, d(M), d(m), m, c>	<COMMIT, **v'**, n, d(M), d(m), m, c>
	<COMMIT, v, **n'**, d(M), d(m), m, c>
<CHECKPOINT, n, d(M)>	<CHECKPOINT, **n'**, d(M)>
	<CHECKPOINT, n, d(**M'**)>
<VIEW-CHANGE, v, P, Q, R>	<VIEW-CHANGE, **v'**, P, Q, R>
	<VIEW-CHANGE, v, **P'**, Q, R>
	<VIEW-CHANGE, v, P, **Q'**, R>
	<VIEW-CHANGE, v, P, Q, **R'**>
<NEW-VIEW, v, V, X, M>	<NEW-VIEW, **v'**, V, X, M>
	<NEW-VIEW, v, **V'**, X, M>
	<NEW-VIEW, v, V, **X'**, M>
	<NEW-VIEW, v, V, X, **M'**>

Figure 1. Structure aware mutations implemented for hBFT.

4. Results

		Agreement		Liveness	
		ss	as	ss	as
N = 0	P = 1	1	0	0	0
N = 0	P = 2	2	1	0	0
N = 1	P = 1	1	0	0	0
N = 1	P = 2	1	0	0	0
N = 2	P = 1	0	0	0	0
N = 2	P = 2	1	1	0	0

Drop Message Weight	Mutate Message Weight	Agreement	Liveness
0	0	1	0
0	25	0	0
0	50	1	0
25	25	0	0
25	50	0	0
50	25	0	0
50	50	0	0

Figure 2. Results of ByzzFuzz (left) and baseline (right) of testing hBFT.

		Agreement		Liveness	
		ss	as	ss	as
N = 0	P = 1	79	1	0	0
N = 0	P = 2	126	3	0	0
N = 1	P = 1	61	1	0	0
N = 1	P = 2	97	6	0	0
N = 2	P = 1	47	0	0	0
N = 2	P = 2	74	4	0	0

Drop Message Weight	Mutate Message Weight	Agreement	Liveness
0	25	5	0
0	50	8	0
25	25	2	0
25	50	4	0
50	25	1	0
50	50	1	0

Figure 3. Results of ByzzFuzz (left) and baseline (right) of the bug injected version of hBFT.

		Agreement	
		Small Scope- "sync"	Small Scope- "async"
N = 0	P = 1	0	0
N = 0	P = 2	0	2
N = 1	P = 1	0	0
N = 1	P = 2	3	6

Figure 4. Results of ByzzFuzz in the controlled (forced) environment of reproducing the known bug.

5. Conclusion

- ByzzFuzz found a potential violation, an injected bug, and under controlled environment a known violation.
- ByzzFuzz is effective at discovering bugs in the implementation of hBFT.
- ByzzFuzz is more effective than baseline methods.
- Small-scope mutations are better at finding bugs than any-scope mutations.

6. Limitations

- Due to the high number of mutations, it is hard to discover the known bug, which would require a higher number of scenarios.
- Our implementation of ByzzFuzz does not cover “bounded-liveness”.
- Our implementation of hBFT might be different from the paper in some aspects, thus any bugs found are specific to our implementation.

References

[1] Sisi Duan, Sean Peisert, and Karl N. Levitt. hbft: Speculative byzantine fault tolerance with minimum cost. IEEE Transactions on Dependable and Secure Computing, 12(1):58–70, 2015.
[2] Levin N. Winter, Florena Buse, Daan de Graaf, Klaus von Gleissenthall, and Burcu Kulahcioglu Ozkan. Randomized testing of byzantine fault tolerant algorithms.7(OOPSLA1), April 2023.