Using Weighted Voting to Accelerate Blockchain Consensus

Artur Brodovic | A.Brodovic@student.tudelft.nl

Introduction

WHEAT[2] optimization allows achieving sub-second speed up for regular Blockchain algorithms. AWARE[1] extends it by implementing a dynamic voting weight allocation algorithm. However, it is vulnerable to performance degradation attacks. This research analyses weak points of AWARE and presents a new Reflection algorithm that makes latency matrix more reliable.

Research questions

- How fake reports affect AWARE dynamic vote allocation?
- What is a behaviour of AWARE dynamic leader selection algorithm during attacks?
- How to use consensus message arrival times to improve latency estimation?

Attacks on performance

Byzantine nodes can act correctly during latency measurement stage and reduce performance after obtaining V_{max} voting weights.
Such attack can increase quorum size. Fig 1. Shows a network that can make fast progress only in a small quorum of 5 nodes.

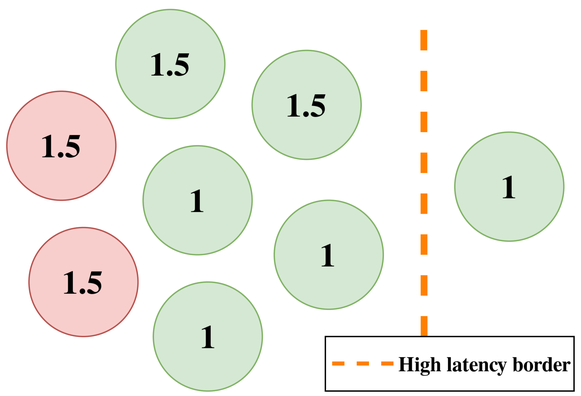

Fig. 1: Network vulnurable to throughput degradation attack

Approach

Reflection algorithm tracks ACCEPT, WRITE and PROPOSE message arrival time from other nodes (see Fig 2.). Later, it sanitizes reported latency vector $\vec{L}$ in such a way that it could explain waiting times between messages.

We are creating a separate Linux Network Namespace for every node to isolate it. Linux Traffic Control (TC) subsystem allows us to introduce queuing discipline (qdisc) that simulates latency between nodes (see Fig 3.).

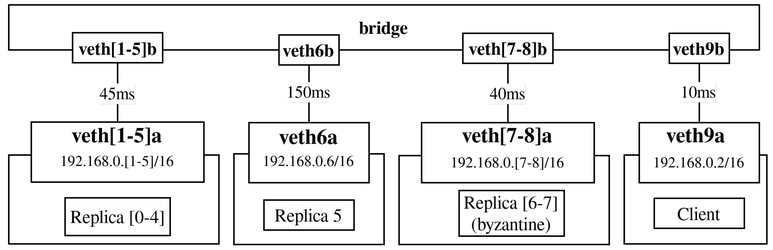

Fig. 3: Latency simulation

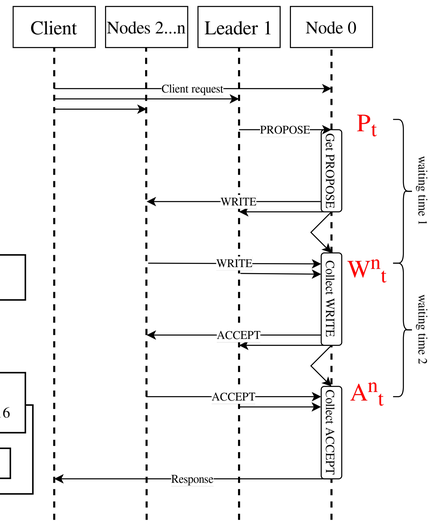

Fig. 2: Waiting for messages

Experiment results

To validate Reflection algorithm, we simulated a network from Fig 1 with parameters $f = 2$ and $\Delta = 1$. So, to advance blockchain, correct replicas need to get $Q_v = 7$ voting weights. That means only if 5 well-connected correct replicas obtain all V_{max} weights, then Blockchain can make fast progress. We can see how Byzantine nodes slow down the network. It explains why AWARE attacked executes transactions slower than AWARE normal (see Fig 6.) However, we can see that after few epochs of working in a slow configuration reconfiguration happens (see Figures 4, 5) and Reflection algorithm reallocates voting weights of Byzantine nodes to correct nodes.

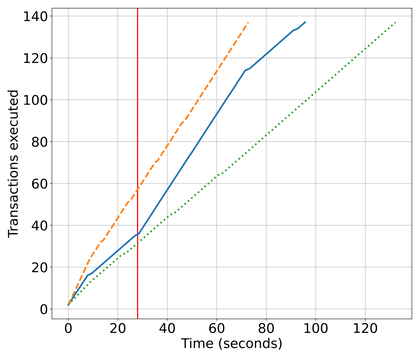

Fig. 4: Drop all messages

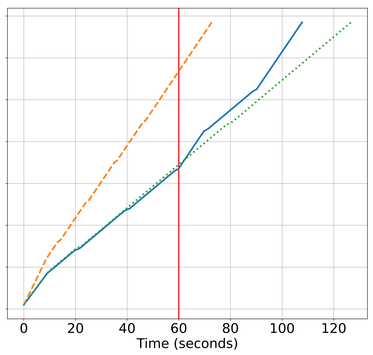

Fig. 5: Drop only ACCEPT messages

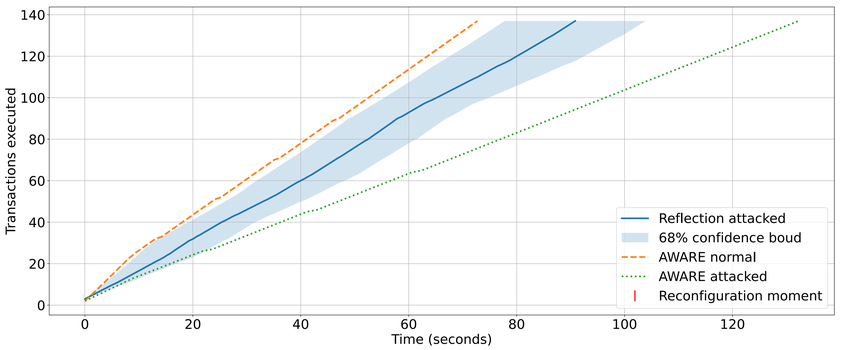

Fig. 6: Drop all messages Reflection average performance over 8 runs

Discussion

BFT-SMaRt implements circular queue (see Fig 7) to choose a new leader. However, AWARE adds a separate leader switch system that does not have an order and works in parallel. As intrusion tolerant systems can not rely on one leader for a long time, they mitigate benefits of AWARE.

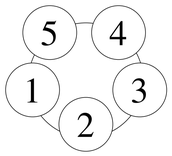

Fig. 7: Ciruclar queue

Conclusion

- Our approach adapts the dynamic link latency estimation algorithm to use information about consensus message arrival times.
- It is better to disable AWARE dynamic leader selection optimization in a network where attacks are expected.

This research shows how to make AWARE optimizations more resilient to Byzantine node attacks.

Acknowledgements

Author: Artur Brodovic
Responsible Professor: Jérémie Decouchan
Supervisor: Rowdy Chotkan

References

[1] Christian Berger, Hans P. Reiser, João Sousa, and Alysson Bessani. Aware: Adaptive wide-area replication for fast and resilient byzantine consensus. IEEE Transactions on Dependable and Secure Computing, 19(3):1605–1620, 2020.

[2] João Sousa and Alysson Bessani. Separating the wheat from the chaff: An empirical design for geo-replicated state machines. 34th IEEE Symp. on Reliable Distributed Systems (SRDS), pages 146–155, 09 2015.