Enhancing DAG-Based Consensus Protocols with Weighted Voting: A Performance Analysis of Narwhal and Tusk

Vian Robotin (V.Robotin@student.tudelft.nl)

Supervised by Dr. Jérémie Decouchant and Rowdy Chotkan

TU Delft

1. Background

Consensus algorithms typically have n nodes and can tolerate up to f faulty nodes.

WHEAT [4]

- introduces a weight assignment scheme, designed to rely on the fastest replicas in the system
- proposes the addition of Δ additional nodes
- allocates $1 + \frac{\Delta}{f}$ weight to the 2f fastest nodes in the system and weight 1 to the rest.
- changes the threshold for quorum formation from $2 * f + 1$ to $2 * (f + \Delta) + 1$

AWARE [3]

- self-optimization framework for BFT protocols
- provides a method to predict the latency of PBFT given a set of network latencies
- detects changes in the network and redistributes the weights to achieve optimal consensus latency

Narwhal and Tusk [1]

- DAG-based consensus algorithm
- Narwhal is responsible for the dissemination of the blocks in the network and building a DAG representing the causal history
- validators propose blocks after receiving $2 * f + 1$ certificates
- validators create certificates when $2 * f + 1$ validators have signed the proposed block.
- Tusk operates on the resulting DAG and makes sure that nodes totally order the DAG.

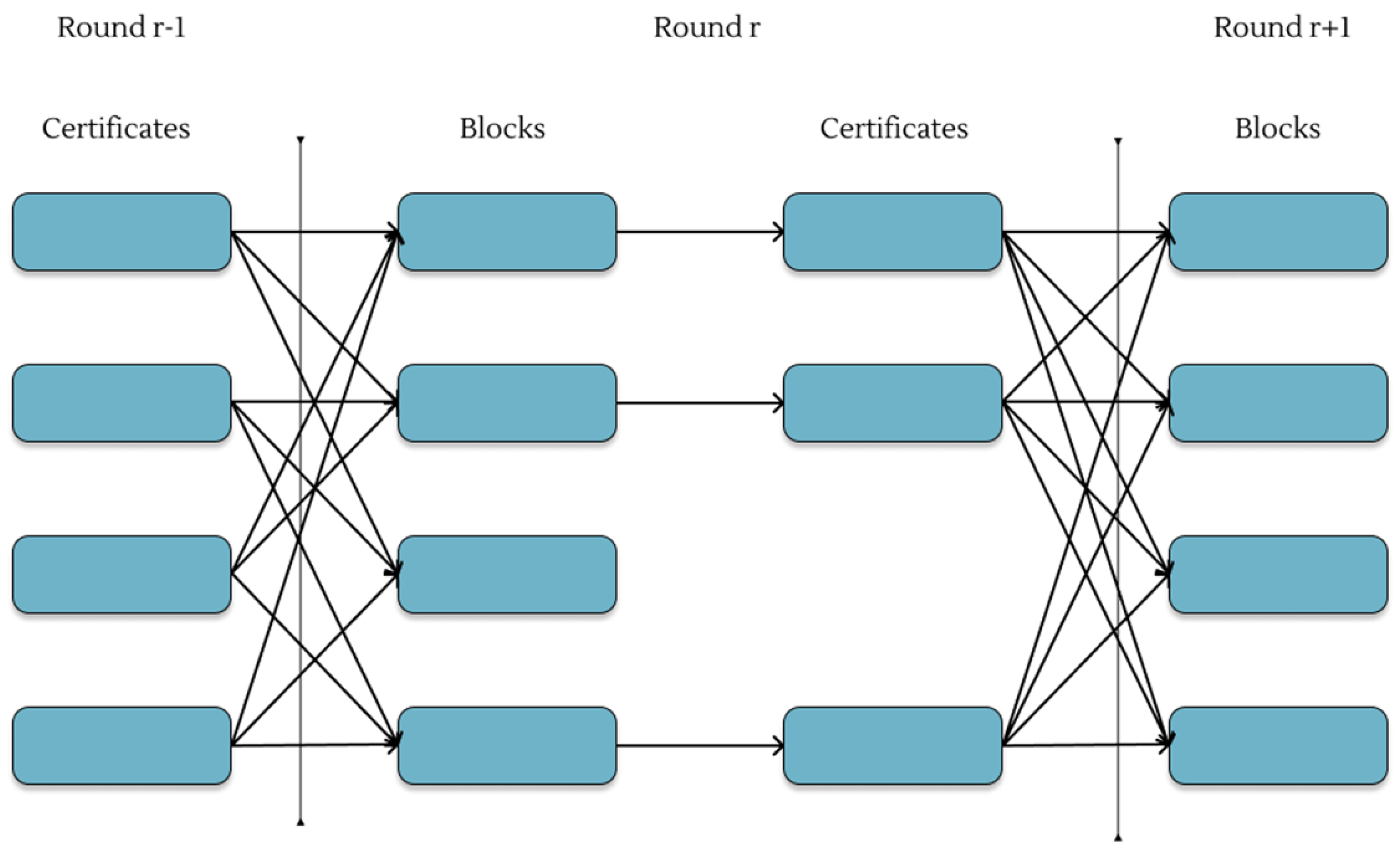

Figure 1: Three rounds of Narwhal.

2. Research Question

How can weighted voting improve the performance of Narwhal and Tusk?

3. Methodology

Gathered real-world latency measurements from CloudPing.

Weighted Narwhal:

- implemented latency prediction model to estimate round completion times based on weighted votes.
- introduced Δ additional validators and assigned weights.
- defined four objective functions (Mean, Max, Stddev, Mean+Stddev) to optimize consensus latency.
- utilized Exhaustive Search and Simulated Annealing for optimal weight assignment.

Weighted Narwhal and Tusk:

- modified original Narwhal and Tusk
- introduced artificial latencies to simulate CloudPing data.
- adjusted quorum formation thresholds
- assigned weights based on the model.

4. Results

Figure 2: Narwhal performance metrics (n = 11, f = 3).

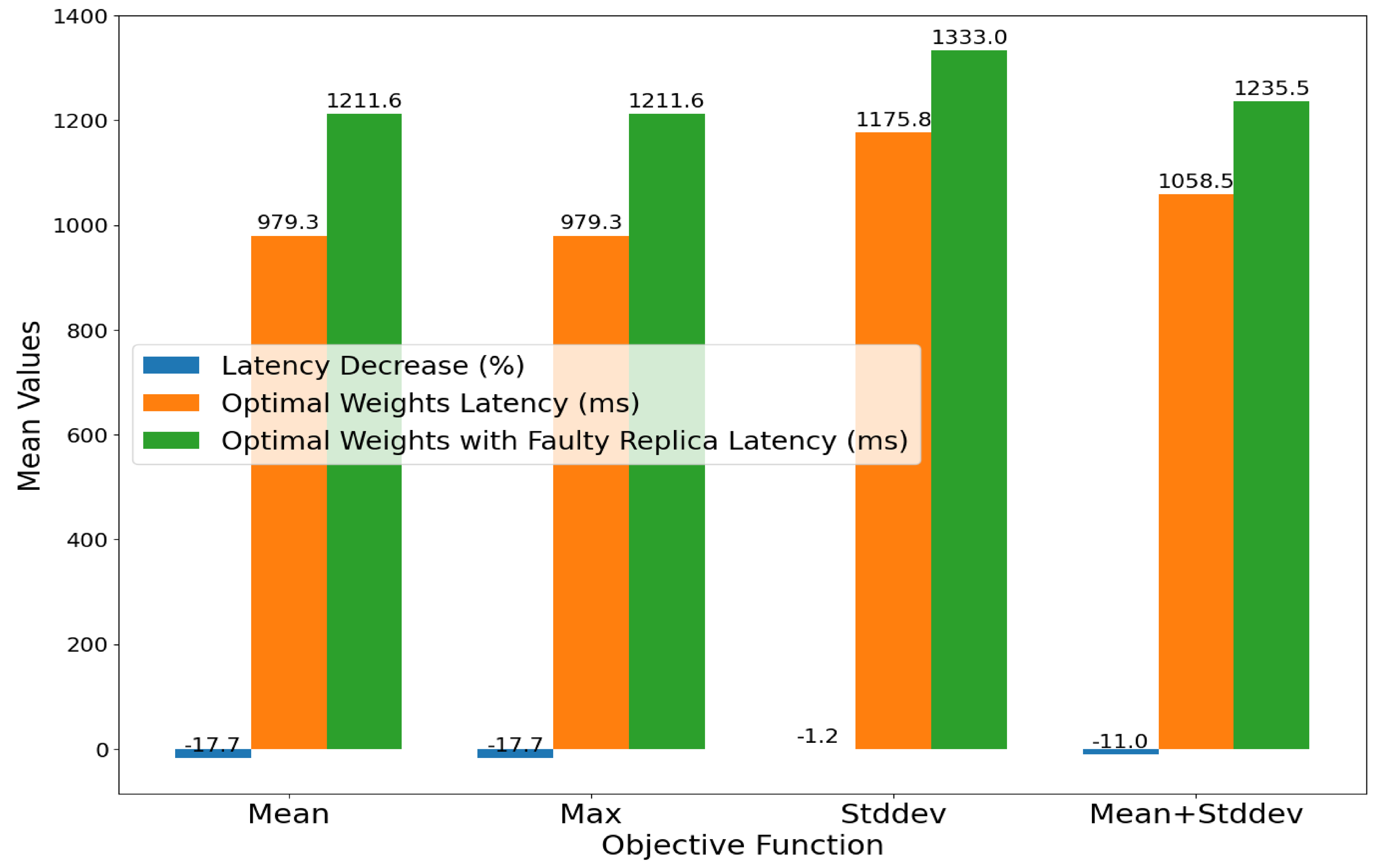

Table 1: Narwhal and Tusk evaluation metrics with different weight assignments.

	Unweighted	Mean	Max	Stddev
Consensus TPS	36,296 tx/s	27,330 tx/s	29,394 tx/s	21,572 tx/s
Consensus BPS	18,583,673 B/s	13,993,181 B/s	15,049,521 B/s	11,044,740 B/s
Consensus latency	2,607 ms	1,673 ms	2,008 ms	2,245 ms
End-to-end TPS	35,977 tx/s	27,086 tx/s	29,151 tx/s	20,675 tx/s
End-to-end BPS	18,420,032 B/s	13,868,152 B/s	14,925,111 B/s	10,585,449 B/s
End-to-end latency	2,982 ms	1,961 ms	2,643 ms	2,788 ms

5. Conclusion

- Weighted voting can significantly reduce consensus latency in both Narwhal and Tusk.
- The Mean objective function achieved the lowest consensus and end-to-end latency.
- Limitation: the search algorithms are computationally expensive

References

[1] George Danezis, Eleftherios Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. Proceedings of the Seventeenth European Conference on Computer Systems, 2021.

[2] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In 3rd Symposium on Operating Systems Design and Implementation (OSDI 99), New Orleans, LA, February 1999. USENIX Association.

[3] Christian Berger, Hans P Reiser, Joao 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.

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