WeightCast: Accelerating Atomic Multicast with Weighted Voting

Author: Stanisław Malinowski
Responsible Professor: Jérémie Decouchant
Delft University of Technology
CSE3000 Research Project Q4 2025/2026

TU Delft
Faculty of Eleectrical Engineering, Mathematics and Computer Science
s.p.malinowski@student.tudelft.nl

1. Motivation

- Genuine atomic multicast globally orders messages, destined to a subset of groups, in a distributed system.
- These systems are often geo-distributed, so their dominant cost is delivery latency.
- A lineage of protocols, have been reducing this latency, measured in the number of communication steps.
- This metric ignores the variability within a step.
- To complete a step, a quorum of votes needs to be collected. Quorum completes, when it's slowest member arrives.
- Weighted voting reduces quorum formation latency, by allowing fast replicas to form smaller quorums.
- So far it has only been applied in leader-based state machine replication (SMR), never in atomic multicast.

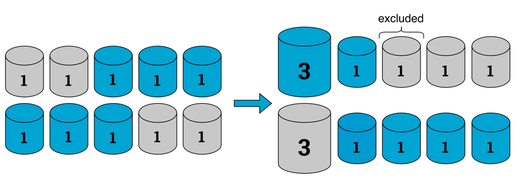
2. Research Questions

How to use weighted voting to accelerate crash-fault-tolerant atomic multicast?

Sub-questions:

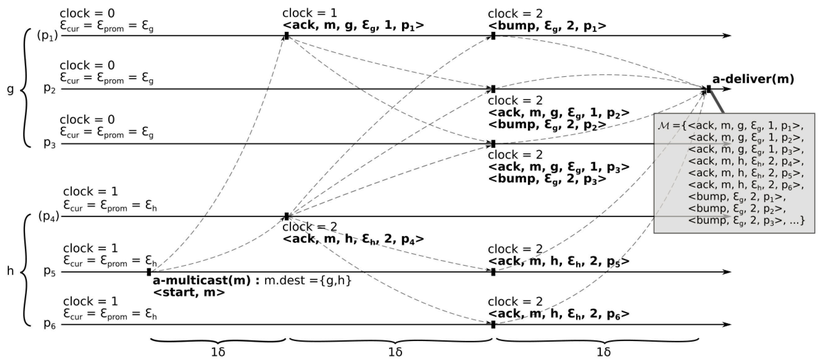
- **RQ1**: How can replica weights be assigned within each destination group so that the message delivery latency in the system is decreased?
- **RQ2**: Does replacing uniform majority quorums with weighted quorums preserve the correctness (total order, agreement, validity) of the overall protocol, and under what conditions?
- **RQ3**: What is the trade-off between the number of tolerated failures and the achievable latency reduction?
- **RQ4**: Under which network topologies and system architectures does the weighted variant outperform the baseline, and by how much?

3. Weighted Quorums



Higher weights form smaller quorums.
→ Fast replicas get high weights.
→ Quorums close faster.

4. PrimCast - Fastest Atomic Multicast



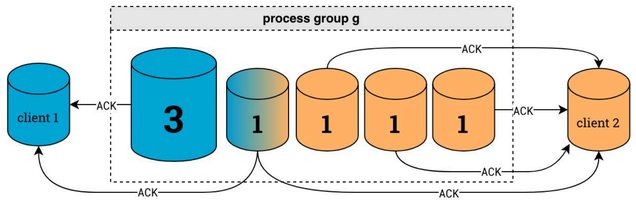
Only messages contributing to delivery at p_2 are shown. Figure from PrimCast paper, by Pacheco et al.

Clients attach to replicas to send messages. Latency is time between a-multicast and a-delivery. Every replica collects votes. Delivery of message m at p is possible when four conditions are met:

- *FINAL-TS* → quorum of ACKs from every destination group has been collected at p
- *PRIMARY-MIN-CLOCK* → p sees its primary's clock >= final timestamp of m
- *QUORUM-CLOCK* → p sees a quorum of its peers' clocks >= final timestamp of m
- *ORDERING* → all messages with possibly lower timestamps have been delivered

Only *FINAL-TS* and *QUORUM-CLOCK* respond to weighted voting. We choose to optimise *FINAL-TS* because it's a predictable and a prerequisite for all other delivery conditions.

5. WeightCast - PrimCast with Weights



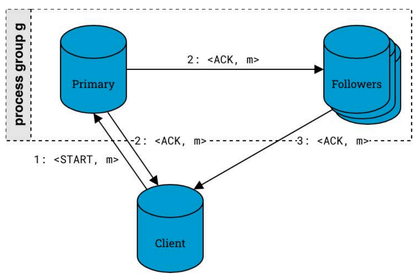
Antagonism: one assignment for all clients → accelerates one at the expense of another.

Structural observation:

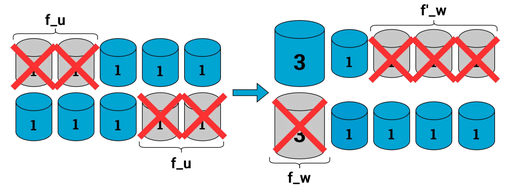
If the *triangle inequality* holds, the primary votes first in every quorum.
→ Weighting the primary is not *antagonistic*

We develop a weight assignment scheme where only the primary gets a high weight.
→ All clients form minimal quorums, and get the latency speedup.

Protocol safety retained because weighted quorums maintain quorum properties.



6. Latency for Fault Tolerance

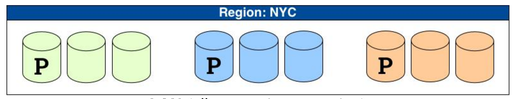


f_w < f_u → r = f_u - f_w
Excluding **r** replicas for **r** less *worst-case* fault tolerance

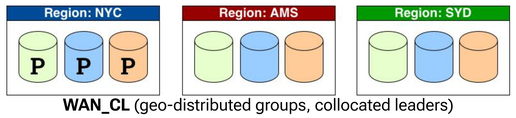
7. Methodology and Results

Discrete event simulation: system of **7x7** groups, **7** regions. Uniform random workload. 5% latency jitter.
Swept parameters:
r: { 0 (unweighted), 1, 2 }
msg_density: { 1,1000 }
clock_type: hybrid clock on and off
topology: { LAN, WAN_CL, WAN_DL }

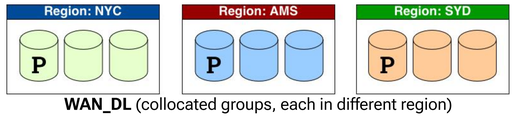
Topologies (different colors are different process groups)**:**



LAN (all groups in one region)

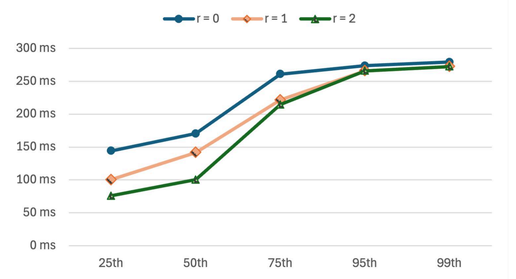


WAN_CL (geo-distributed groups, collocated leaders)



WAN_DL (collocated groups, each in different region)

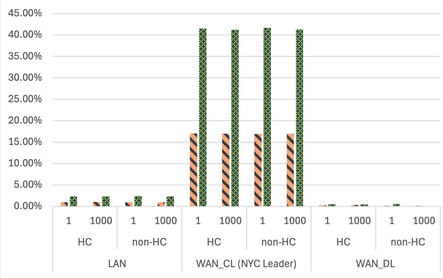
Results:



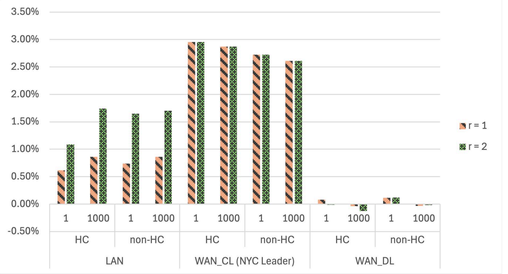
WAN_CL delivery latency for **r** in {0,1,2}, across percentiles. Graph shape similar regardless of other parameters.

8. Results (contd.)

Relative latency reduction over **r**=0, across swept params.



50th percentile



95th percentile

9. Conclusions

- **RQ1**: Heavy weight to primary, low to followers.
- **RQ2**: **Yes**, assignment maintains quorum properties.
- **RQ3**: **Less fault tolerance → lower latency** (with diminishing returns.
- **RQ4**: In systems where groups are internally geo-distributed → messaging latencies between members are high. Relative **latency improvement** of **up to 50%** (25th percentile). Improvements do **not affect tails**.

10. Limitations, Future Work

- Evaluation on simulator → real deployment to verify.
- Only one delivery condition targeted
  → real-time empirical optimisation could help further reduce latency. Both increasing the gain ceiling from increasing r, and pushing tails down.
- Triangle inequality can be violated in a WAN
  → real-time empirical optimisation could help.