Raft With RAPID

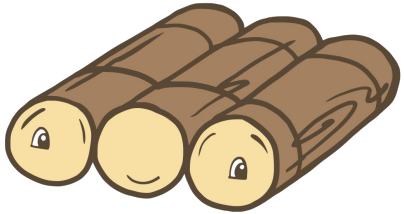
Creating a Simpler Consensus Algorithm

Presenters: Alex Chandler, David Klinger

12-04-2023

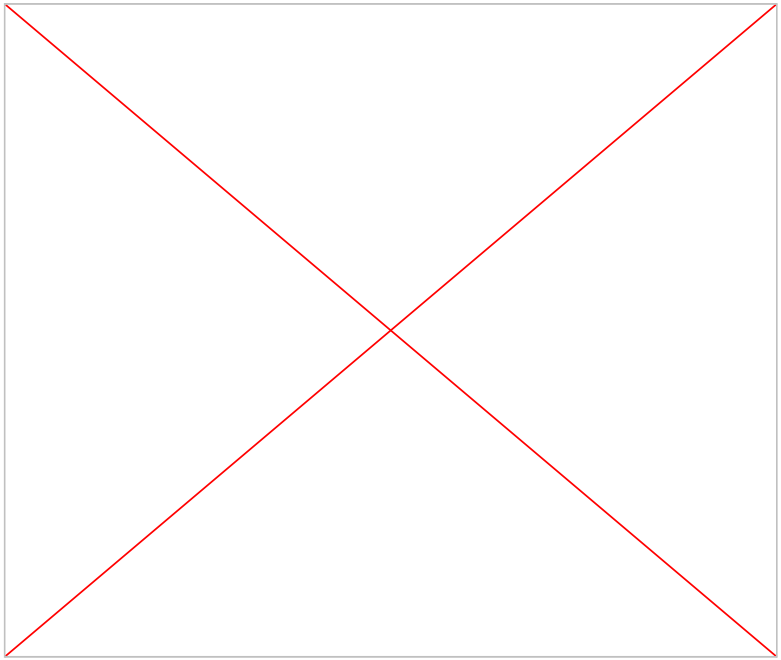
Motivation

- In Search of A Simpler Consensus Algorithm
- Why Build off of Raft?
    - Designed for Simplicity and Understandability:
        - Breaks down consensus into manageable subproblems.
        - More straightforward to learn, implement, and debug compared to complex algorithms like Paxos.
    - We want to make Raft Simpler
    - Widely Used with Strong Community Support:
        - Gained popularity for its clarity in handling consensus.
        - Extensive documentation, open-source implementations, and support across various programming languages.



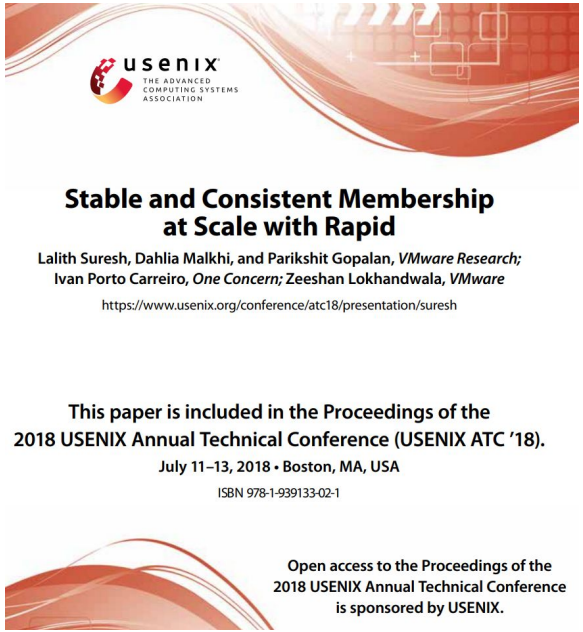
Raft Overview

- Leader-Centric Model:
  - One leader manages log replication; followers accept entries.
  - Leader sends heartbeats to followers.
- Node States and Transitions:
  - Followers become Candidates if no heartbeat from leader.
- Leader Election with Randomized Timeouts:
  - Triggered when a follower's randomized timeout elapses.
  - Candidates request votes; wins leadership with majority.
- Terms and Consistency:
  - The cluster operates in terms of numbered “terms”.
  - Only one leader allowed per term to ensure consistency.
- Log Contents and Role:
  - Logs contain client commands and configuration changes.

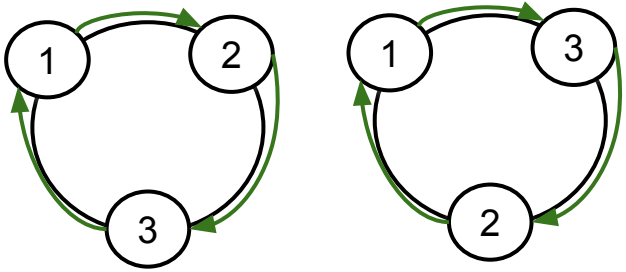


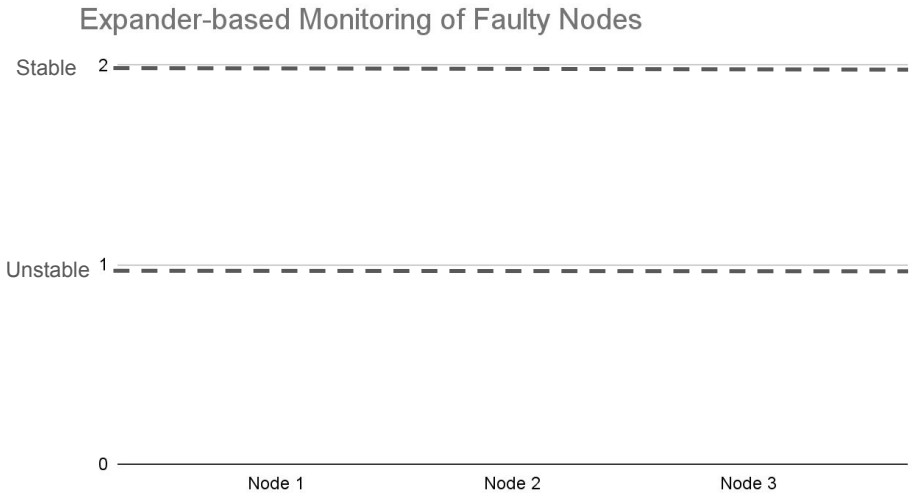
RAPID Overview

- Why Use RAPID:
  - RAPID provides Stable and Consistent Membership Views.
- Multi-Process Cut Detection (CD):
  - Provides stable and accurate failure detection by requiring multiple confirmations for suspecting process faults.
- Leaderless Consensus Protocol:
  - Multi-Process Cut Detection ensures almost everywhere agreement.
  - Requires only one round of communication to decide the next configuration.
- Scalability and Speed:
  - Offers 2-5.8x faster cluster bootstrapping than Memberlist and ZooKeeper, while ensuring consistency and stability in large-scale deployments.
- Integration and Versatility:
  - Easily integrates into existing distributed applications.
  - Proven effectiveness in a range of enterprise data center environments and failure scenarios.

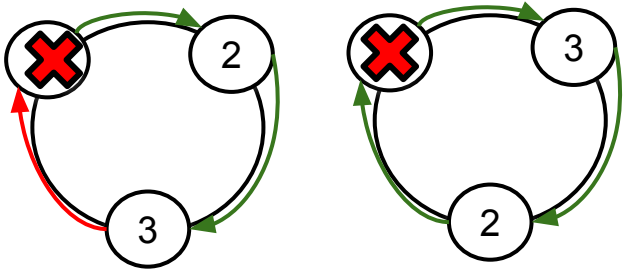


RAPID Fault Detection Mechanism

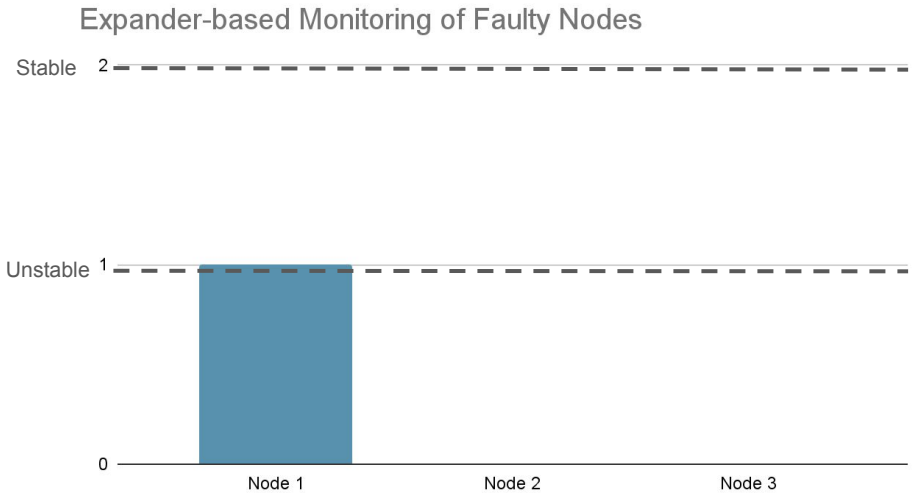




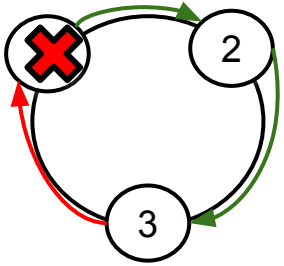
RAPID Fault Detection Mechanism

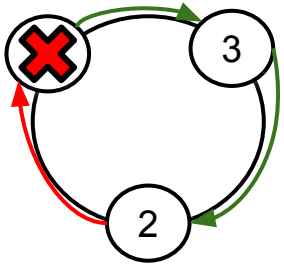


3 realizes first that 1 is not responding

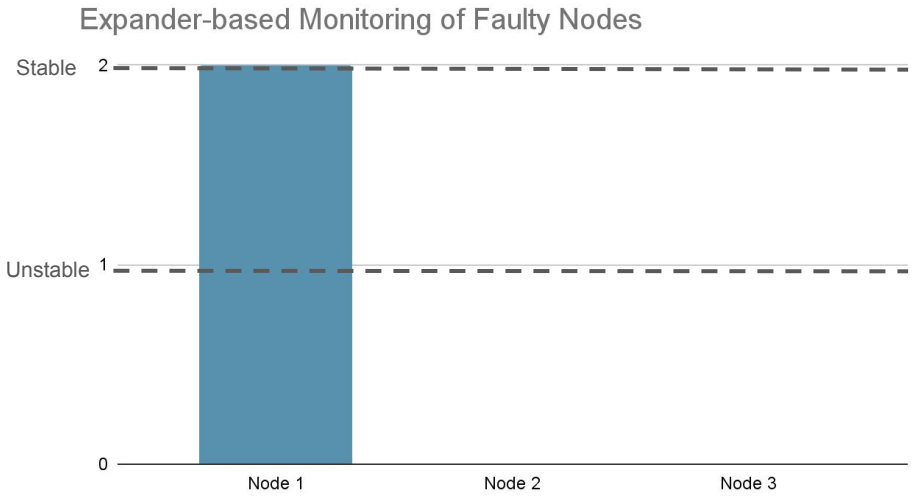


RAPID Fault Detection Mechanism





2 then realizes that 1 is not responding



Advantages of RAPID in Raft: How RAPID simplifies and enhances Raft

- Simplification of Cluster Management:
  - RAPID assumes responsibility for tracking active nodes and managing cluster configurations, reducing the complexity of Raft's replicated log.
  - No longer need two overlapping cluster configurations before: (old→old_new→new) after: (old→new).
- Simplifies Leader Implementation:
  - Simplifies the leader's role by eliminating the need for continuous heartbeat messages, focusing only on proposing new log entries.
- Simplifies Log Implementation:
  - Raft stores both client values and cluster configuration changes in the log.
  - In Raft with RAPID, the replicated log now only stores client-proposed entries.

How Integrating RAPID Simplifies Cluster Configuration Changes

Before: Raft Configuration Change

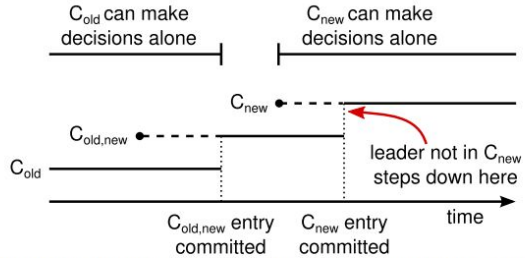
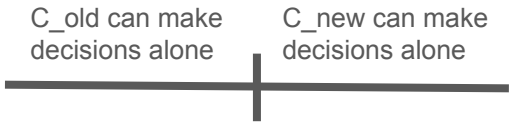


Figure 11: Timeline for a configuration change. Dashed lines show configuration entries that have been created but not committed, and solid lines show the latest committed configuration entry. The leader first creates the $C_{old,new}$ configuration entry in its log and commits it to $C_{old,new}$ (a majority of C_{old} and a majority of C_{new}). Then it creates the C_{new} entry and commits it to a majority of C_{new}. There is no point in time in which C_{old} and C_{new} can both make decisions independently.

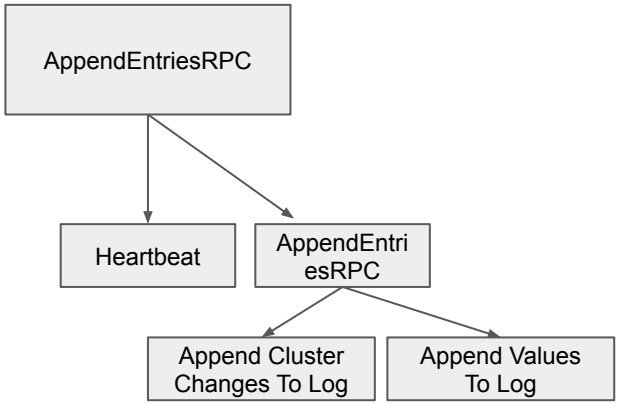
After: Raft-RAPID Configuration Change



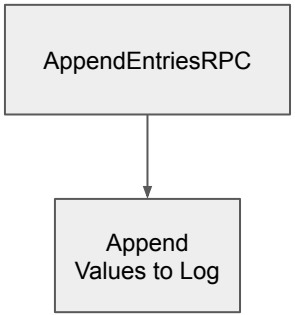
Limitation: The replication factor does not change

How Raft-RAPID Simplifies Handing of Append Entries RPC

Raft Append Entries



Raft-RAPID Append Entries

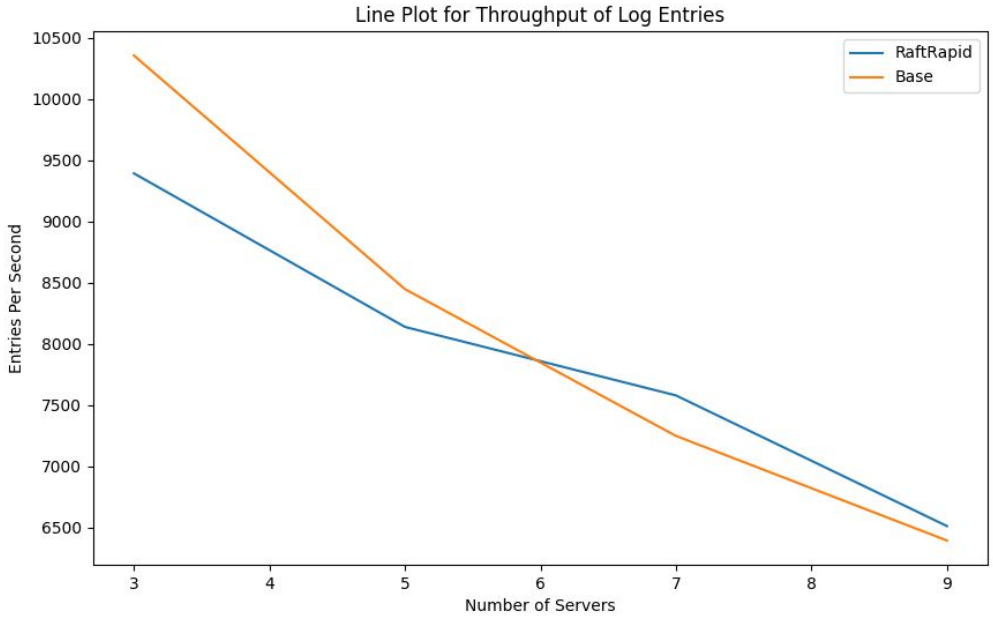


Evaluation

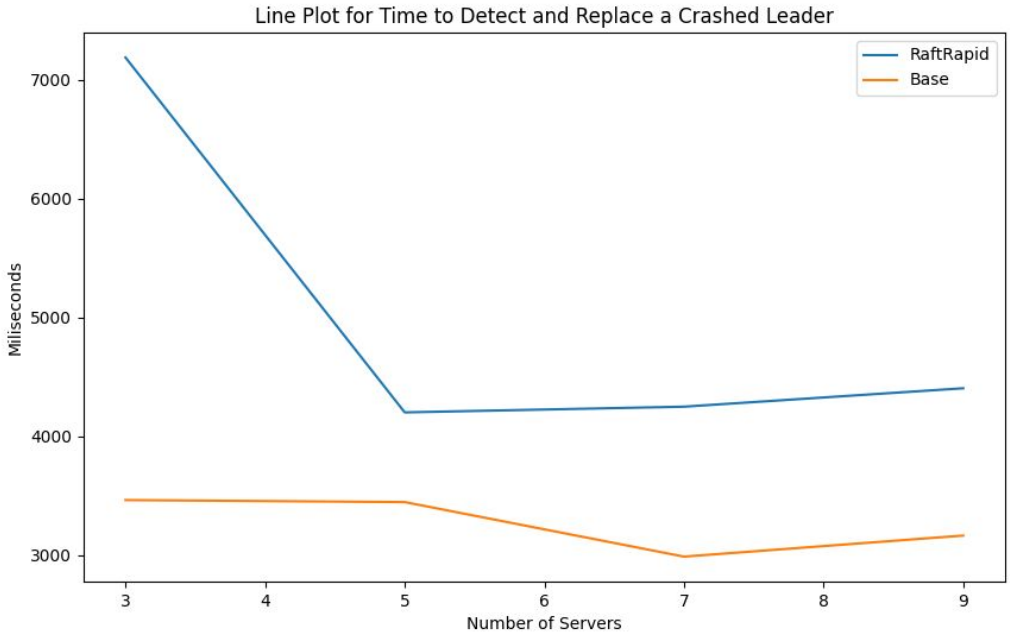
We compare Raft to Raft with RAPID along the following

1. Throughput of log entries
2. Time to detect and replace a crashed leader
3. Time from network partition healing to elected leader
4. Memory utilization
5. Network usage
    a. Bandwidth
    b. Latency
6. Configuration change time

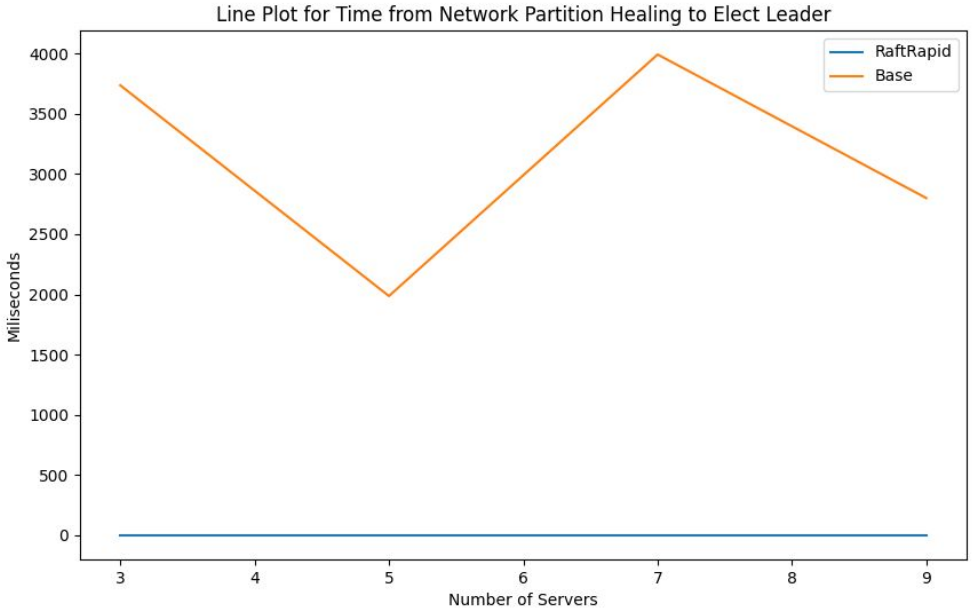
Throughput of Log Entries



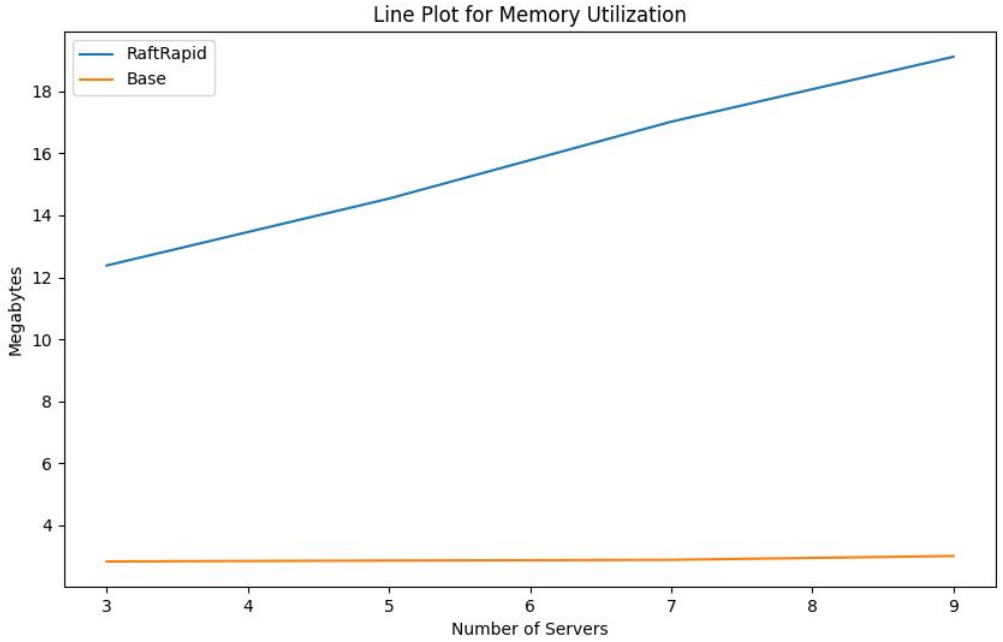
Time to Detect and Replace a Crashed Leader



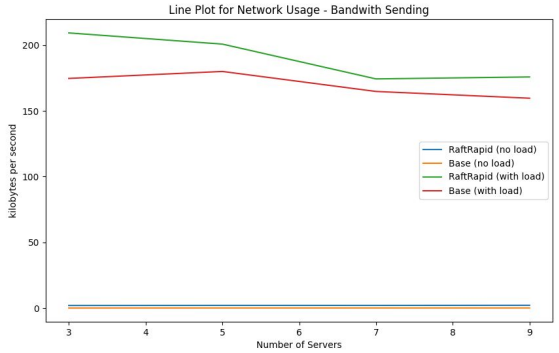
Time from Network Partition Healing to Elected Leader

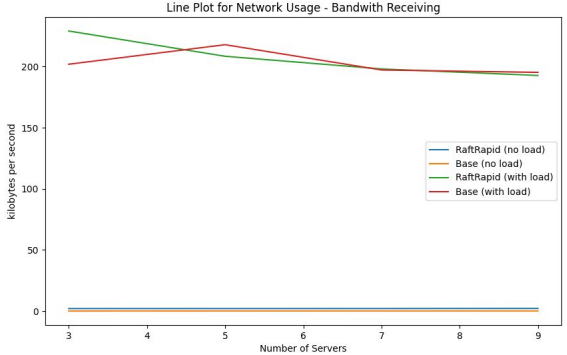


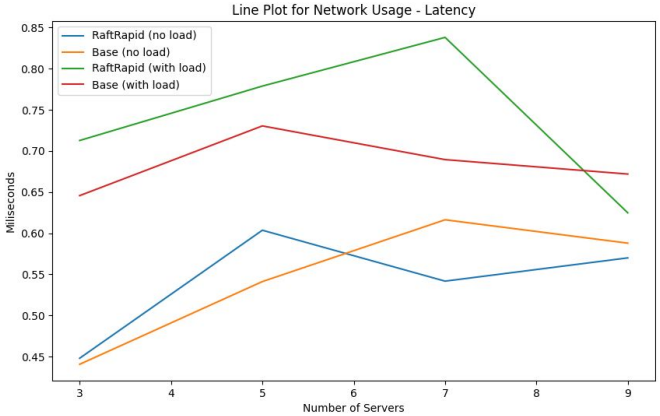
Memory Utilization



Network Usage







Survey

Link in chat for voting Raft or Raft-RAPID:
Also available here: https://forms.gle/1y2ZCaT1m3vydans5

Thank You!