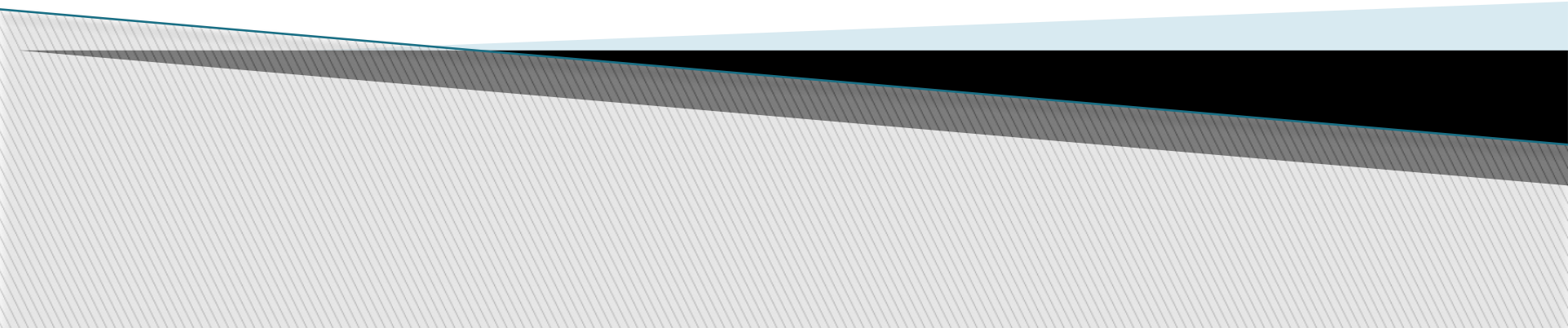
A Gossip Based Approach for Failure Detection and System State Consensus

Josh Clemm



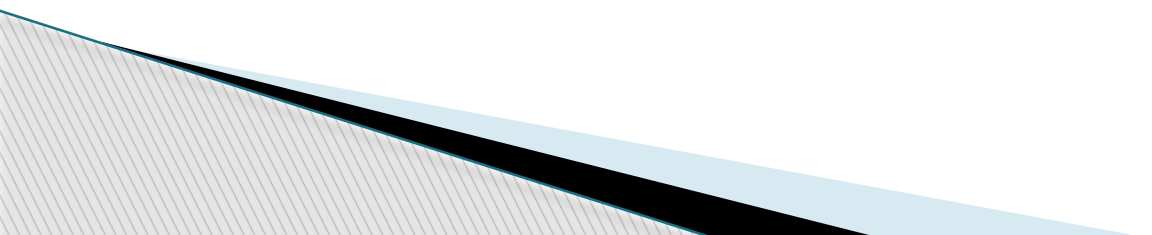
Agenda

- Gossip protocol
  - Overview
  - Implementation
  - Performance
- Personal prototyping
- Redesigned approach using Gossip
  - Overview
  - Diagrams
  - SLOC estimates
  - Bandwidth estimates

Overview

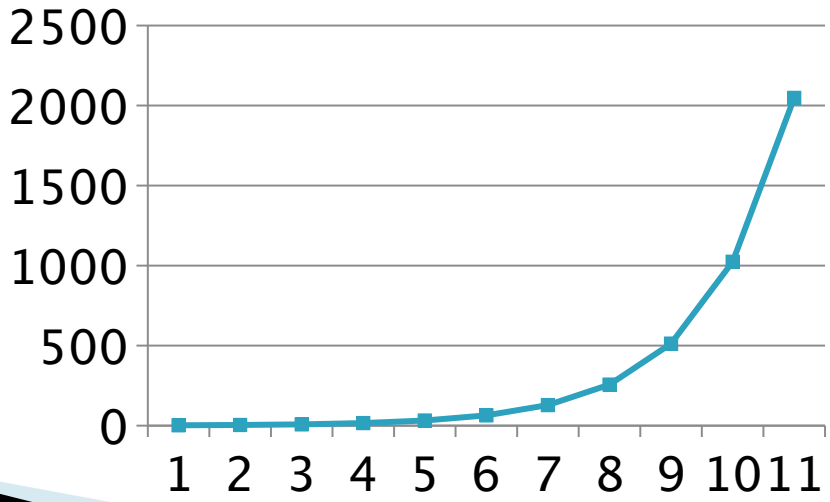
- In distributed systems, you must detect failure asynchronously because of arbitrarily long network delays
- Failure detectors must be able to differentiate between a dead process and a slow process that is alive
- Processes must communicate amongst themselves to achieve consensus on the state of all nodes in the system
- Agreement can be achieved using group communication and gossiping, among others.

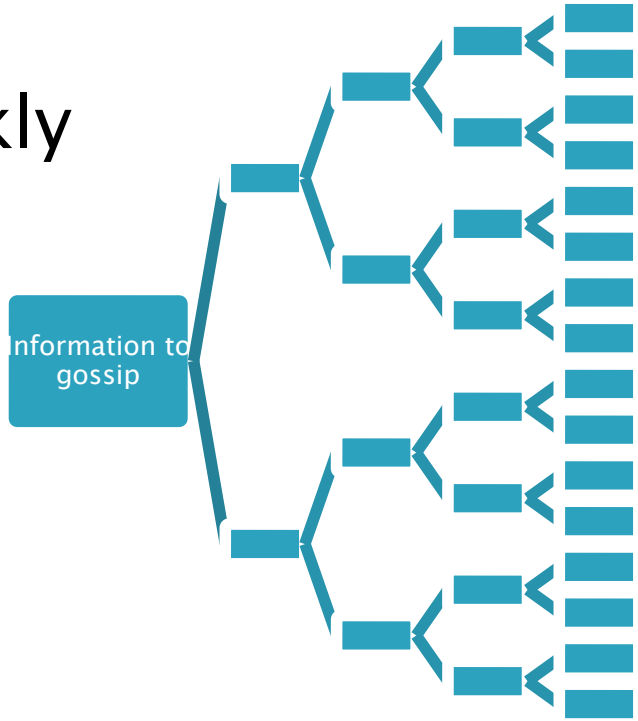
Gossip Protocol Overview



Gossip Protocol Overview

- Nodes individually gossip their state, and the states of other nodes, to one another to eventually achieve a unified view of the system.
- Information propagates quickly





Gossip implementation overview

- Very basic
- Heartbeat messages are the foundation
- Each member maintains a list containing each known member and a heartbeat counter for each member
- This list is sent and received by all members

Implementation details

- Each member in the system has two main responsibilities:
  - Active cycle
    - Every t_gossip ms, update own heartbeat counter (increment), and send own membership list to random member.
  - Passive cycle
    - Asynchronously receive a neighbor message, and merge their list with your list
    - Increasing heartbeats from other members signify liveliness
- Finally, a node is removed from own list if a period of t_cleanup ms has passed without an updated heartbeat

Gossip Performance

- Agreement times based on gossip frequency:

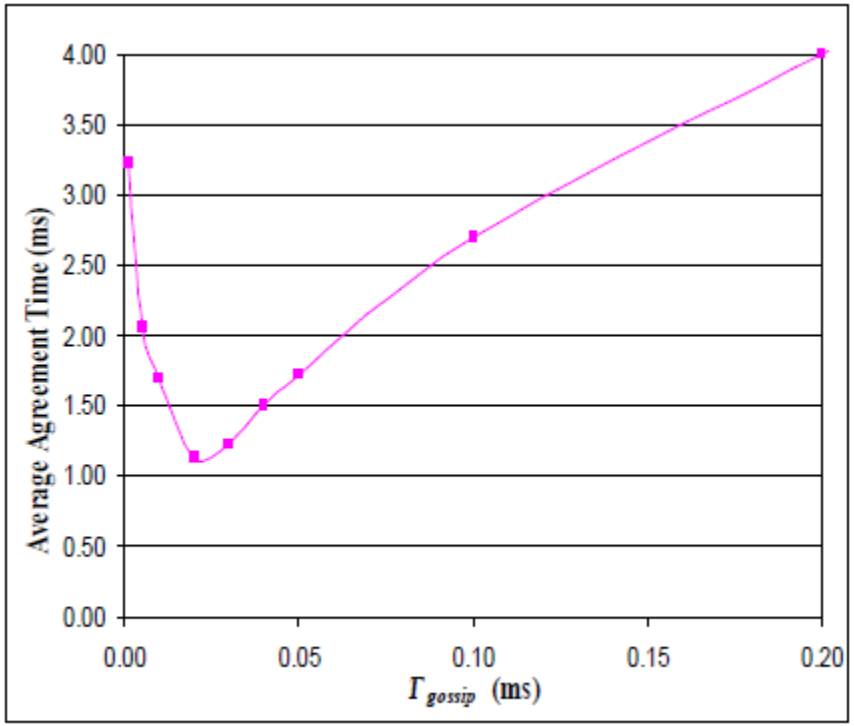


Fig. 7. Time to reach agreement for various gossiping intervals.

This graph reflects 96 nodes. For small values of Tgossip, the time to reach agreement increases due to the congestion imposed by excessive gossiping on the network.

For large values of Tgossip, the time to reach agreement is large because the gossip list information is not propagating from node to node at an efficient rate.

Gossip Performance

- Agreement times based on timeout frequency:

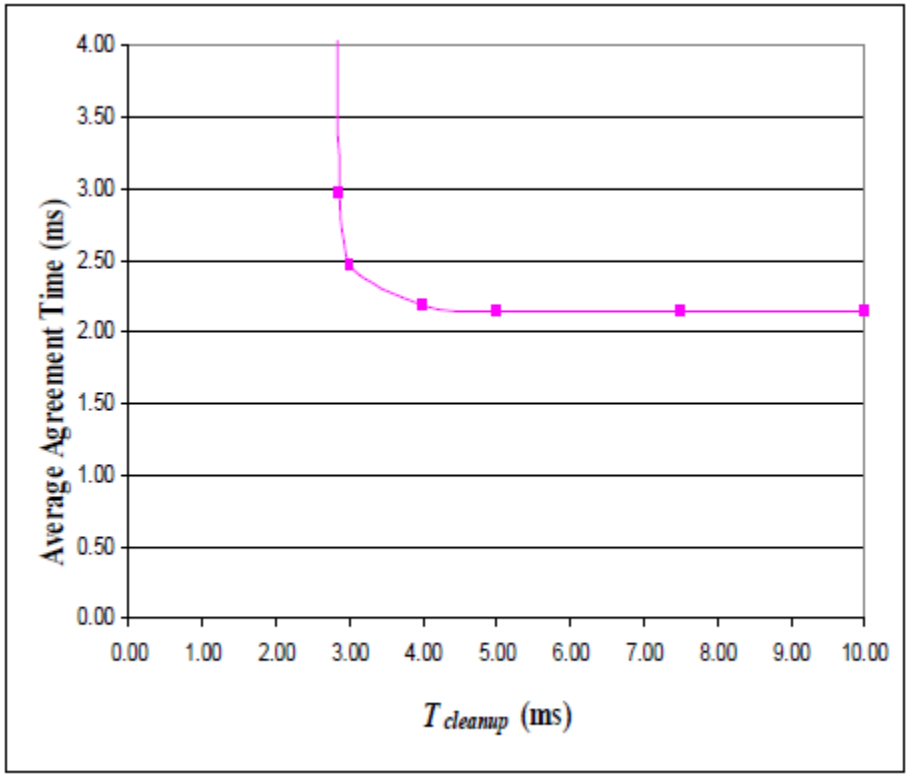


Fig. 8. Agreement time versus $T_{cleanup}$ for the 96-node system; T_{gossip} = 0.1ms.

For small values of Tcleanup, the time to reach agreement is large due to false failure detections; a false failure detection occurs when a node is incorrectly purged from another node's gossip list even though the purged node is still alive.

For larger values of Tcleanup, false failure detections are minimized, and the time to reach agreement becomes relatively constant.

Gossip Performance

- Agreement times on network partitions:

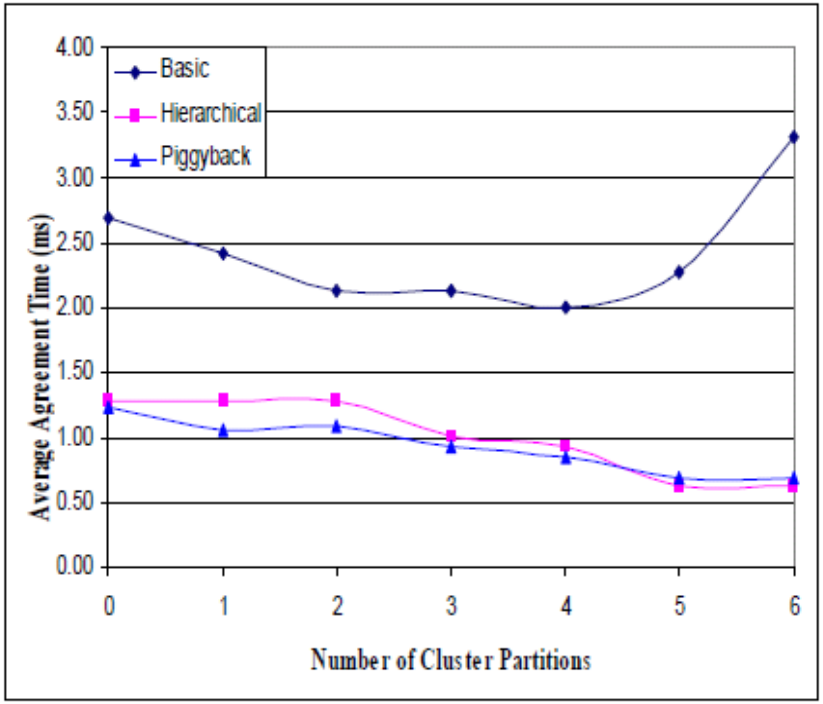


Fig. 14. Protocol agreement time for network partitions; T_{gossip} = 0.1ms, $T_{cleanup}$ = 10ms, L = 0.9 and P_{ucast} = 0.25

With the basic protocol, the agreement time initially decreases as the number of partitions increases because the largest group of nodes that must agree becomes smaller. However, as the number of partitions increases further, the percentage of gossips sent to unreachable nodes increases.

The basic protocol becomes less efficient as more partitions occur, and therefore the agreement time increases

Gossip benefits/drawbacks

- Benefits
  - Simple – two parameters to adjust
  - Scalable
  - Handles network partitions and failures very well
  - Uses less overhead than reliable multicast (groups)
- Drawbacks
  - Consensus is achieved less rapidly than with a broadcast or multicast approach (groups).
  - Requires frequent gossip messages

Personal Experimentation

- Easy to implement
- Successful in reaching consensus, determining failure
- Can maintain consensus even with 75% packet loss
- Gossip message size:
  - IP address – 4 bytes, heartbeat – 2 bytes, optional port – 4 bytes = 6–10 bytes per node