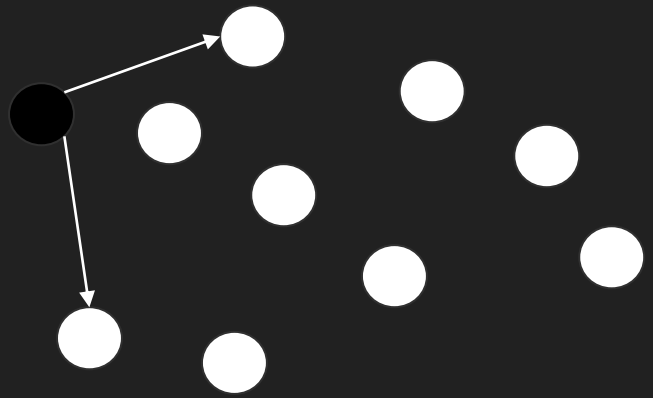
Gossip Protocol

Background

In many systems, we want to be able to pass information around from node to node in order to achieve a broadcasting of information. This can be useful in many types of clusters, but the reason I’m covering it here is because Cassandra uses a gossip protocol in order to have nodes communicate with one another which nodes they believe are active and which have failed. Very useful when trying to communicate across many nodes, without using some sort of centralized coordination service which requires extra infrastructure.

Gossip Protocol Visualized

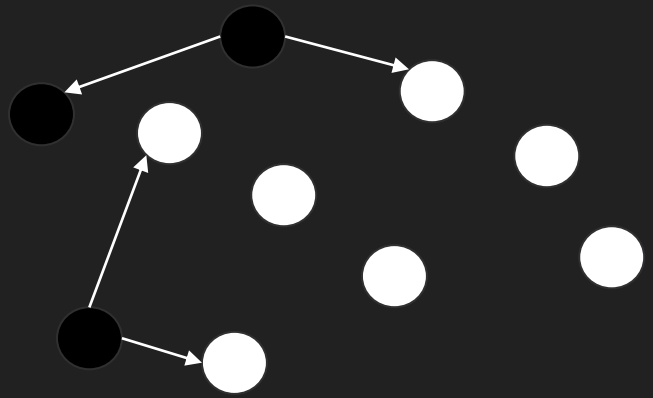
Pick a few (let’s say 2) nodes randomly, and broadcast the message to them!



After a few rounds, there is a high probability each node will receive a message!

Gossip Protocol Visualized

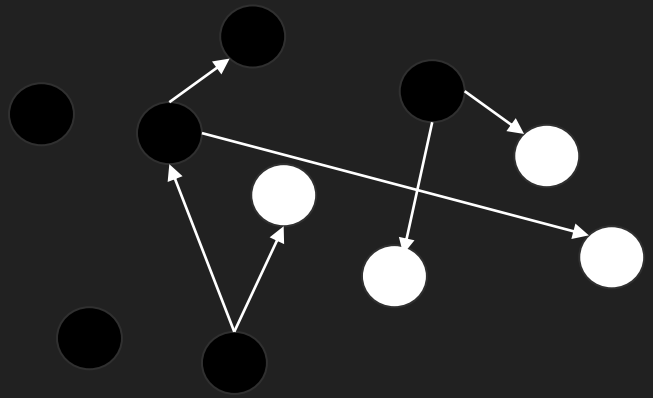
Pick a few (let’s say 2) nodes randomly, and broadcast the message to them!



After a few rounds, there is a high probability each node will receive a message!

Gossip Protocol Visualized

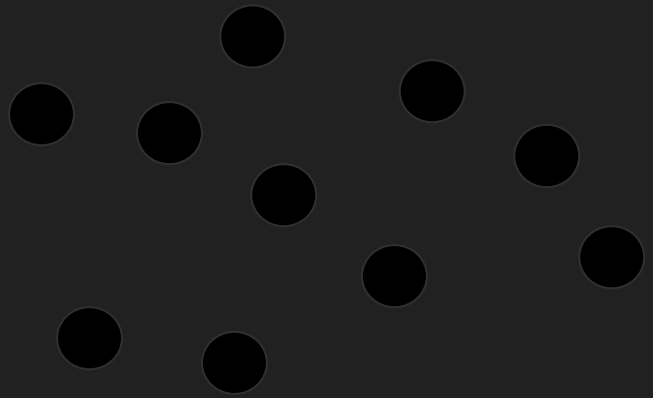
Pick a few (let’s say 2) nodes randomly, and broadcast the message to them!



After a few rounds, there is a high probability each node will receive a message!

Gossip Protocol Visualized

Pick a few (let’s say 2) nodes randomly, and broadcast the message to them!



After a few rounds, there is a high probability each node will receive a message!

Gossip Protocol in Cassandra

Used to pass details about node membership/failure in a cluster:

- Each node is gossiping with other nodes regarding its own local load
- Each node is gossiping with other nodes about heartbeats it receives from all other nodes, which contain a timestamp
- Other nodes, upon receiving gossip, use the timestamps to keep only the most recent messages and then pass them on
- All nodes use the most recent heartbeat timestamp to determine if other nodes in the cluster are down

Gossip Protocol Conclusion

Good for quickly broadcasting messages to many nodes with relatively low overhead/coordination! Useful in large clusters to perform things like failure detection by gossiping all of the heartbeats that they have received. Note that gossip protocols do not ensure any ordering of messages, they are just throwing as many around as possible - for ordering you need some other way of doing so (e.g. timestamps in Cassandra, or maybe a vector clock).