[Consensus in Distributed system]

What is Consensus in Distributed System?

In a distributed system, multiple computers (known as **nodes**) are mutually connected with each other and collaborate with each other through message passing. Now, during computation, they need to agree upon a common value to coordinate among multiple processes. This phenomenon is known as **Distributed Consensus.**

[Gossip protocol]

What Is Gossip Protocol?

The typical problems in a distributed system are the following [1], [11]:

- maintaining the system state (liveness of nodes)
- communication between nodes

The potential solutions to these problems are as follows [1]:

- centralized state management service
- peer-to-peer state management service

Centralized State Management Service

A centralized state management service such as Apache Zookeeper can be configured as the service discovery to keep track of the state of every node in the system. Although this approach provides a strong consistency guarantee, the primary drawbacks are the state management service becomes a single point of failure and runs into scalability problems for a large distributed system [1], [11].

Peer-To-Peer State Management Service

The peer-to-peer state management approach is inclined towards high availability and eventual consistency. The gossip protocol algorithms can be used to implement peer-to-peer state management services with high scalability and improved resilience [1].

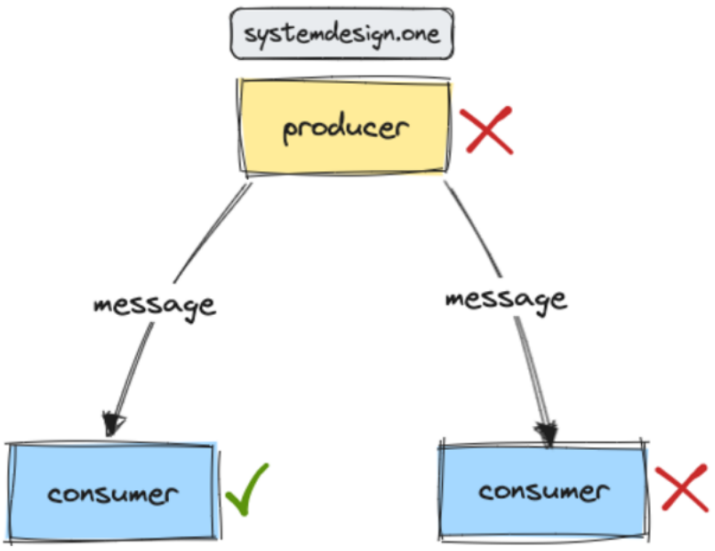
The gossip protocol is also known as the **epidemic protocol** because the transmission of the messages is similar to the way how epidemics spread. The concept of communication in gossip protocol is analogous to the spread of rumors among the office staff or the dissemination of information on a social media website [4], [8].

Broadcast Protocols

The popular message broadcasting techniques in a distributed system are the following:

- point-to-point broadcast
- eager reliable broadcast
- gossip protocol

Point-To-Point Broadcast



The producer sends a message directly to the consumers in a point-to-point broadcast. The retry mechanism on the producer and deduplication mechanism on the consumers makes the point-to-point broadcast reliable. The messages will be lost when the producer and the consumer fail simultaneously [3].

Eager Reliable Broadcast

Every node re-broadcasts the messages to every other node via reliable network links. This approach provides improved fault tolerance because messages are not lost when both the producer and the consumer fail simultaneously. The message will be re-broadcast by the remaining nodes. The caveats of eager reliable broadcast are the following [3], [8]:

- significant network bandwidth usage due to $O(n^2)$ messages being broadcast for n number of nodes
- sending node can become a bottleneck due to O(n) linear broadcast
- every node stores the list of all the nodes in the system causing increased storage costs

Gossip Protocol

The gossip protocol is a decentralized peer-to-peer communication technique to transmit messages in an enormous distributed system [1], [8]. The key concept of gossip protocol is that every node periodically sends out a message to a subset of other random nodes [8], [2]. The entire system will receive the particular message eventually with a high probability [11], [3]. In layman's terms, the gossip protocol is a technique for nodes to build a global map through limited local interactions [1].

- Apache Cassandra employs the gossip protocol to maintain cluster membership, transfer node metadata (token assignment), repair unread data using Merkle trees, and node failure detection
- Consul utilizes the swim-gossip protocol variant for group membership, leader election, and failure detection of consul agents
- CockroachDB operates the gossip protocol to propagate the node metadata
- Hyperledger Fabric blockchain uses the gossip protocol for group membership and ledger metadata transfer
- Riak utilizes the gossip protocol to transmit consistent hash ring state and node metadata around the cluster
- Amazon S3 uses the gossip protocol to spread server state across the system
- Amazon Dynamo employs the gossip protocol for failure detection, and keeping track of node membership
- Redis cluster uses the gossip protocol to propagate the node metadata
- Bitcoin uses the gossip protocol to spread the nonce value across the mining nodes

Gossip Protocol Use Cases

The gossip protocol is used in a multitude of applications where eventual consistency is favored. The popular applications of the gossip protocol are as follows [8], [5], [4], [7], [12]:

- database replication
- information dissemination
- maintaining cluster membership
- failure detection
- generate aggregations (calculate average, maximum, sum)
- generate overlay networks
- leader election

Gossip Algorithm

The high-level overview of the gossip algorithm is the following [6], [1]:

1. every node maintains a list of the subset of nodes and their metadata
2. gossip to a random live peer node's endpoint periodically
3. every node inspects the received gossip message to merge the highest version number to the local dataset

The heartbeat counter of a node is incremented whenever a particular node participates in the gossip exchange. The node is labeled healthy when the heartbeat counter keeps incrementing. On the other hand, the node is considered to be unhealthy when the heartbeat counter has not changed for an extended period due to a network partition or node failure [1]. The following are the different criteria for peer node selection in the gossip protocol [12]:

- utilize library offered by programming languages such as java.util.random
- interact with the least contacted node
- enforce network-topology-aware interaction