State Machine Replication and Consensus Algorithms

Hui Chen [a]

[a]CUNY Brooklyn College

October 15, 2025

Fault-Tolerance in Distributed Systems

Solve large computational problems: using distributed systems

- Clusters, data centers, clouds
- Thousands of machines

Failures are common: hardware, software, and network failures

- Servers crash
- Messages are lost
- Networks are disconnected

Goal: Make progress in solving the problem despite failures

Solution: Replication

- Example: State machine replication (SMR)

Example Distributed Systems

Consider a key-value store: get(key), put(key, value)

- Replicate across n servers using state machine replication
- Key ingredients: logs and states
- Logs: a log that is written before the operation (Write-Ahead Log), a totally ordered sequence of would-do operations, sent to all servers

time	1	2	3	4
entry	put(a, 1)	put(b, 2)	put(a, 3)	pub(c, 4)

- States:

time	a	b	c
1	1	-	-
2	1	2	-
3	3	2	-
4	3	2	4

Consistency: Linearizability

The system is linearizable if it is possible to order all operations in a sequence such that:

- The order is consistent with the real-time sequence of operations
- Each operation appears to happen instantaneously at some point between its invocation and response

Achievable with fail-stop failures using replicas.

Question: How to maintain consistency across different data replicas in case of failures?

Answer: Consensus algorithms, the majority of replicas agree on the same log entry

Consensus Algorithms

Servers agree on the same value in the presence of failures

- Fail-stop failures.
  - Viewstamped Replication (Oki and Liskov 1988), Paxos (Lamport 1998; Lamport 2001), Raft (Ongaro and Ousterhout 2014)
- Byzantine failures.
  - Proof-of-work (Bitcoin), Proof-of-stake (Ethereum)

Goals: Liveness and Safety

- Liveness. System makes progress in solving the problem despite failures
  - Availability. Responds to client's requests despite failures
- Safety. System runs correctly despite failures
  - Consistency. Replicas agree on the same value as if there were no failures

Trade-Off Between Safety and Liveness

Impossibility of distributed consensus with one faulty process (Fischer, Lynch, and Paterson 1985)

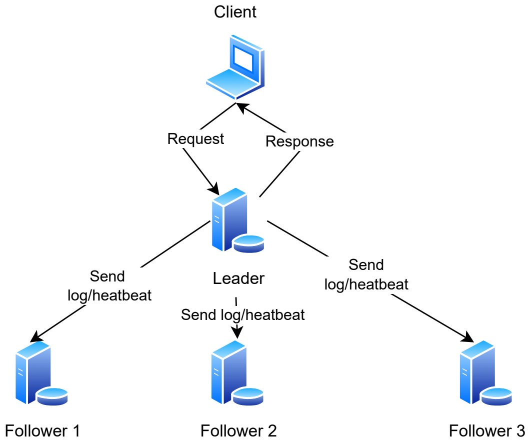
- Theorem: No consensus protocol is totally correct in spite of one fault, i.e., guarantee both safety and liveness in reaching consensus.
- Proof: Because messages can be delayed indefinitely, a consensus algorithm cannot distinguish between a slow node (or network partition) and a failed node.
- Implication: To maintain safety, consensus algorithms must sacrifice liveness under certain failures

Quorum-based consensus: Trade availability for consistency

- Example: Raft (Ongaro and Ousterhout 2014)

Raft Consensus Algorithm: Overview

A quorum-based consensus algorithm



Raft Consensus Algorithm: Overview

- Quorum-based consensus algorithm
- Three components
  - Leader election
  - Log replication
  - Safety
- Leader must be supported by a majority of the group (a quorum)
  - Majority quorum: n servers tolerate f failures, where $n = 2f + 1$
- Availability-consistency trade-off:
  - Consistency. At most one connected subgroup can serve requests
  - Availability. Once a majority of replicas fail, the remaining replicas should not serve requests.

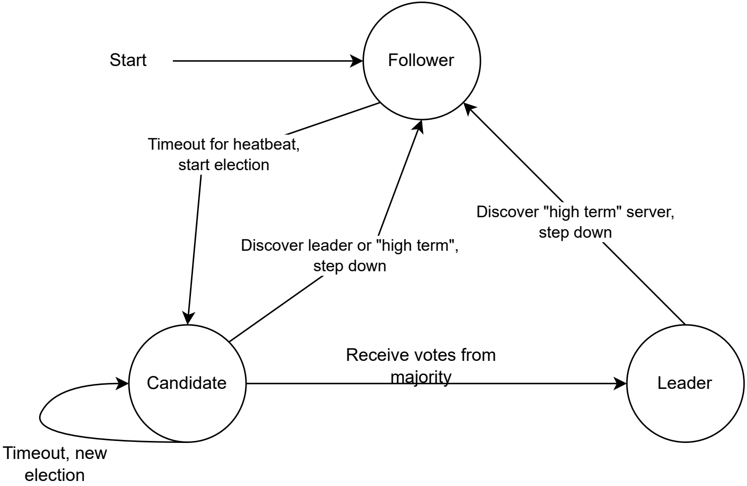
Leader and Followers

The system consists of n servers, each server is either a leader or a follower or a candidate for leader

- The system must always have a single leader in place.
- Only the leader can accept the request sent from clients and responds to the clients.
- The leader is responsible for communicating with all followers
- Normal operation:
  - Prepares WAL, send to followers, apply operations upon receiving responses from a majority of the followers
  - Send a heartbeat to followers to maintain its leadership status; and

Major Components: Leader Election

Which server is the leader? Initially every server is a follower.



Summary

- Fault-tolerance in distributed systems using replication
- Consistency: Linearizability
- Consensus algorithms: Raft
- Trade-off between safety and liveness

Bibliography I

Fischer, Michael J, Nancy A Lynch, and Michael S Paterson (1985). "Impossibility of distributed consensus with one faulty process". In: *Journal of the ACM (JACM)* 32.2, pp. 374–382.

Lamport, Leslie (May 1998). "The part-time parliament". In: *ACM Trans. Comput. Syst.* 16.2, 133–169. ISSN: 0734-2071. DOI: 10.1145/279227.279229. URL: https://doi.org/10.1145/279227.279229.

— (2001). "Paxos made simple". In: *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58.

Oki, Brian M and Barbara H Liskov (1988). "Viewstamped replication: A new primary copy method to support highly-available distributed systems". In: *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pp. 8–17.

Bibliography II

Ongaro, Diego and John Ousterhout (2014). "In search of an understandable consensus algorithm". In: *2014 USENIX annual technical conference (USENIX ATC 14)*, pp. 305–319.