CSCI 5673
Distributed Systems

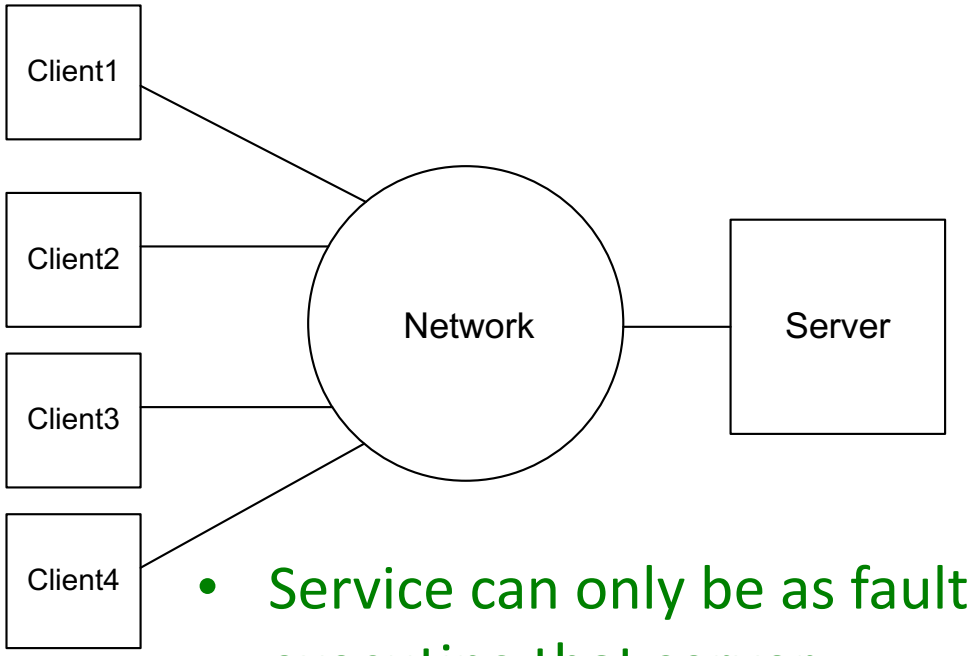
Lecture Set Five

Replicated State Machines

Lecture Notes by
Shivakant Mishra
Computer Science, CU Boulder
Last update: February 21, 2017

F. Schneider. Implementing Fault Tolerant Services Using the State Machine Approach. ACM Computing Surveys, December 1990.

Client/Server Model



- Service is implemented by one or more servers
  - export operations
- Clients invoke operations by sending requests

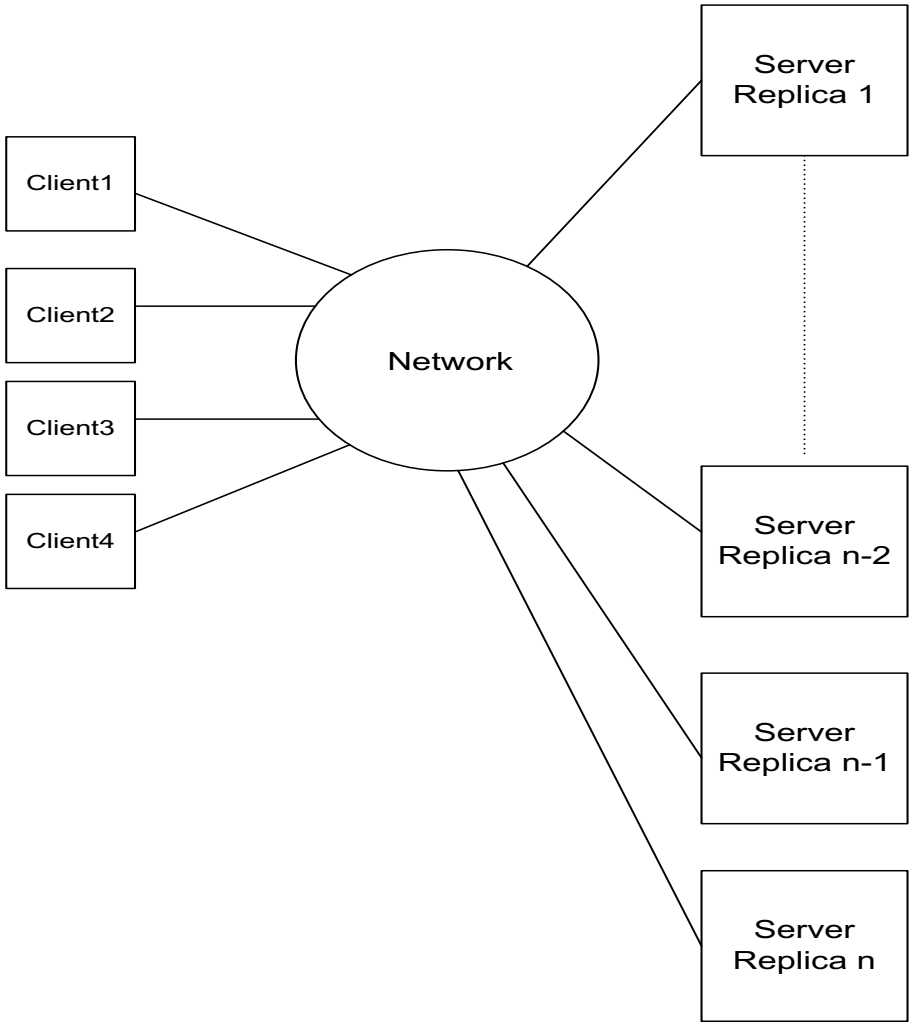
- Service can only be as fault tolerant as the processor executing that server
- To improve the level of fault tolerance, multiple servers that fail independently must be used
- Replicas of a single server are executed on separate processors of a distributed system, and protocols are used to coordinate client interactions with these replicas

State Machine Approach

- The state machine approach is a general method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas
- The approach also provides a framework for understanding and designing replication management protocols
- Viewing protocols that involve replication of data or software in terms of state machines helps in understanding how and why they work

The “State Machine Approach” was introduced by Leslie Lamport in “Time, Clocks and Ordering of Events in Distributed Systems.”

Replication Scheme



State Machine Model

- Each Server Replica is an identical state machine
- State Machines are Request Driven Machines and cannot progress on their own
- A client Issues a Request to the State Machine

State Machines: State variable and deterministic commands

Requests and Causality, Happened Before Relation

- State machines process requests consistent with potentially causality
- Client A sends r, then r'.
- r is processed before r'.
- r causes Client B to send r’’.
- r is processed before r’’.

State Machine Coding

- State Machines are procedures
- Client calls procedure
- Avoid loops
- More flexible structure

Consensus

- Ensures procedures are called in same order across all machines

Tolerating Failures

- t fault tolerant: Correct service is guaranteed as long as
  - ≤ t components are faulty
  - Simply where the guarantees end
- Statistical Measures
  - Mean time between failures
  - Probability of failure over interval
  - other

Failure types: Fail Stop, Crash, Byzantine

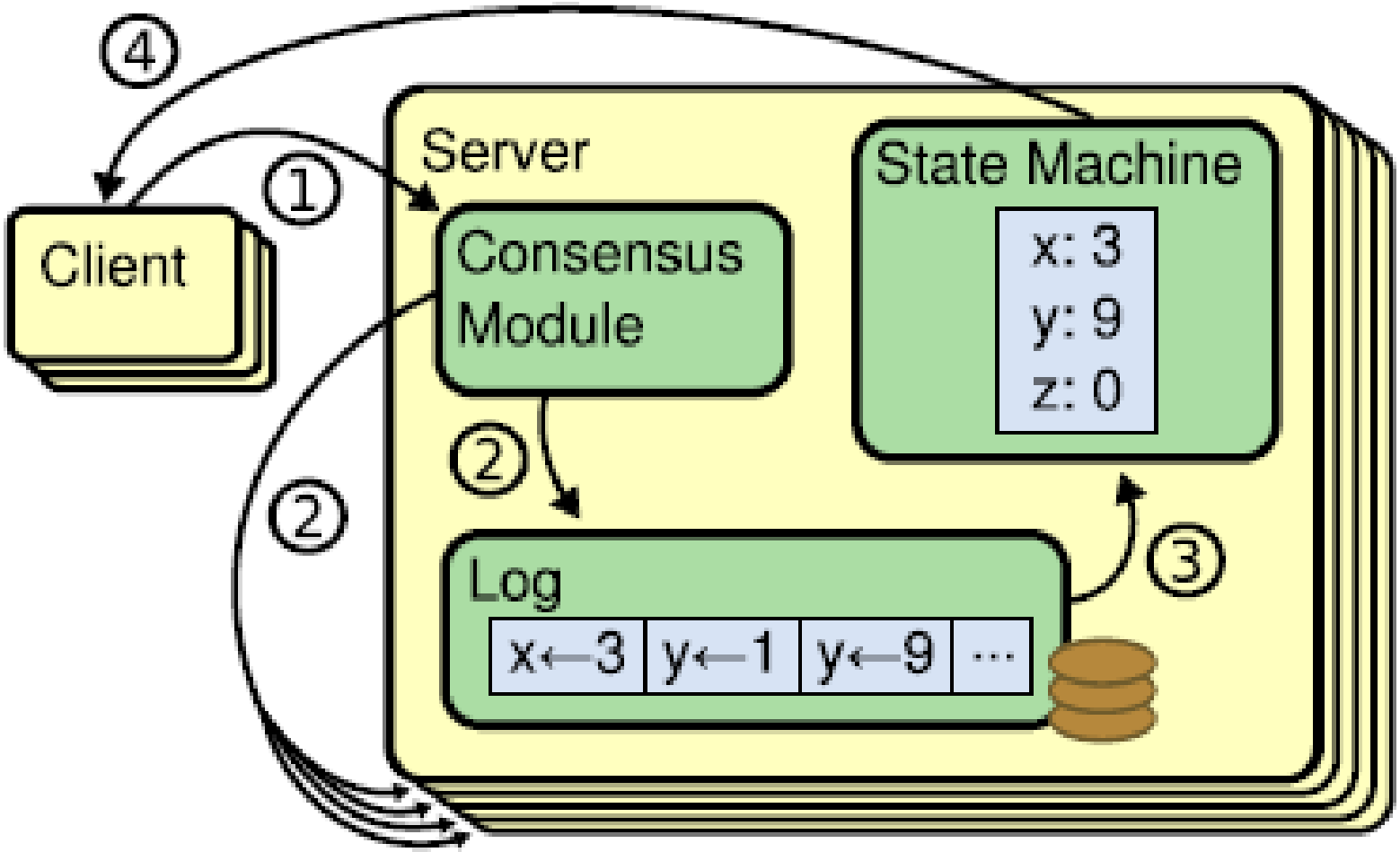
Fault Tolerant State Machines

- Implement the state machine on multiple processors with independent failure modes
- State Machine Replication
  - Each state machine replica starts in the same initial state
  - Executes the same requests
  - Requires consensus to execute in same order
  - Deterministic, each will do the exact same thing
  - Produce the same output

Implementing Replicated State Machines

- Each server maintains a log containing commands
- Consensus algorithm ensures that all logs contain the ***same commands*** in the same order
- State machines always execute commands ***in the log order***
  - They will remain consistent as long as command executions have ***deterministic results***

Implementing Replicated State Machines



Implementing Replicated State Machines

- Client sends a command to one of the servers
- Server adds the command to its log
- Server forwards the new log entry to the other servers
- Once a consensus has been reached, each server state machine process the command and sends a reply to the client

Implementing Replicated State Machines

- Typically satisfy the following properties
  - ***Safety:***
    - Non-faulty servers never return an incorrect result to the client
  - ***Availability:***
    - Remain available as long as a majority of the servers (non Byzantine failures) remain operational and can communicate with each other and with clients

Replicated State Machine: Implementation

- We will discuss three different implemetations
  - Totem
  - Paxos
  - Raft