Distributed Systems -Fault Tolerance

INTRODUCTION

- Distributed Computing Systems consists of variety of hardware and software components.
- Failure of any of these components can lead to unanticipated, potentially disruptive behavior and service unavailability.
- Fault may occur due to various reasons like communication failure, resources or hardware failure, failure due to fault in process, software errors.
- A system's fault tolerant capability guarantees to minimize the loss that may be caused due to the unpredictable system behavior

Fault

- At the lowest level of abstraction fault can be termed as “defect”.
- It can lead to inaccurate system state. Fault in the system can be categorized based on time as below:
  - Transient: This type of fault occurs once and disappear
  - Intermittent: This type of fault occurs many time in an irregular way
  - Permanent: This is the fault that is permanent and brings system to halt

Failure

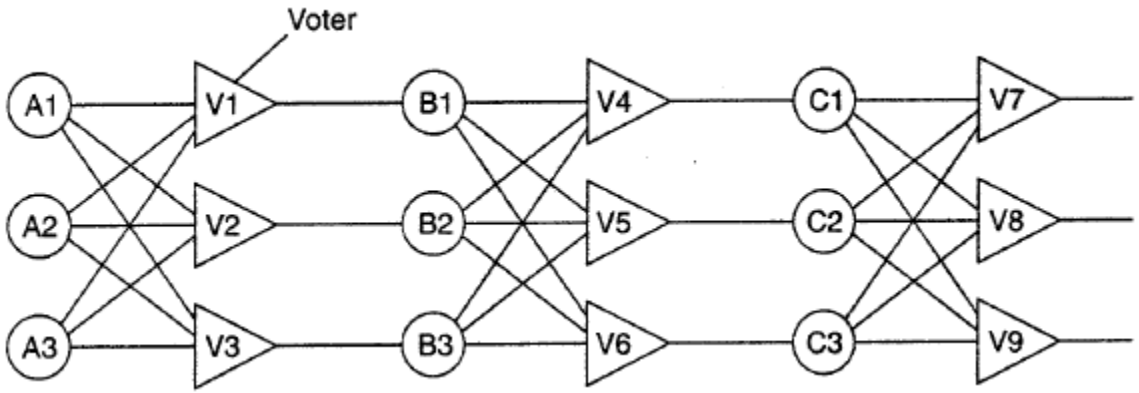
- May be defined as faults due to unintentional intrusion. Different types of failure are as below
  - Crash Failure: A server halts, but is working correctly until it halts
  - Omission Failure: A server fails to respond incoming requests
    - Receive omission: A server fails to respond incoming message
    - Send Omission: A server fails to send message
  - Timing Failure: A server's response lies outside the specified time interval
  - Response Failure: The server's response in incorrect
    - Value failure: The value of the response is wrong
    - State transition failure: The server deviates from the correct flow of control
  - Arbitrary (Byzanthine) Failure: A server may produce arbitrary responses at arbitrary times

Need

- Availability: The system must be usable immediately at any time.
- Reliability: The system must work for a long period of time without error.
- Safety: There should be no catastrophic consequences of temporal failure.
- Maintainability: The system must be able to repair and fix the fault quickly and easily.
- Safety: The system should be able to resist the attacks against its integrity

Failure masking by redundancy

- Information redundancy: Extra bits are added (e.g. CRC)
- Time redundancy: Action may be redone (e.g. transaction after abort)
- Physical redundancy: Hardware and software component may be multiplied (e.g. adding extra disk, replicating the database), TMR
- Triple modular redundancy (TMR)



Metrics

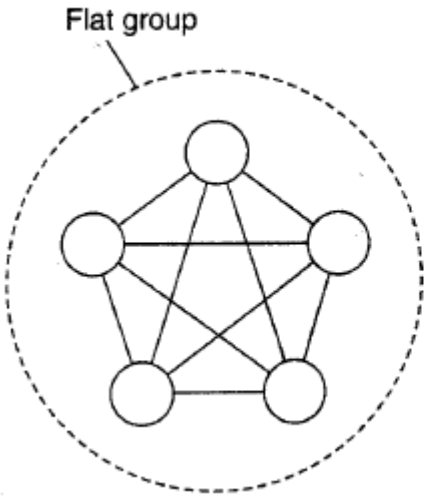
- Mean Time To Failure (MTTF): The average time until a component fails.
- Mean Time To Repair (MTTR): The average time needed to repair a component.
- Mean Time Between Failures (MTBF): Simply MTTF + MTTR.

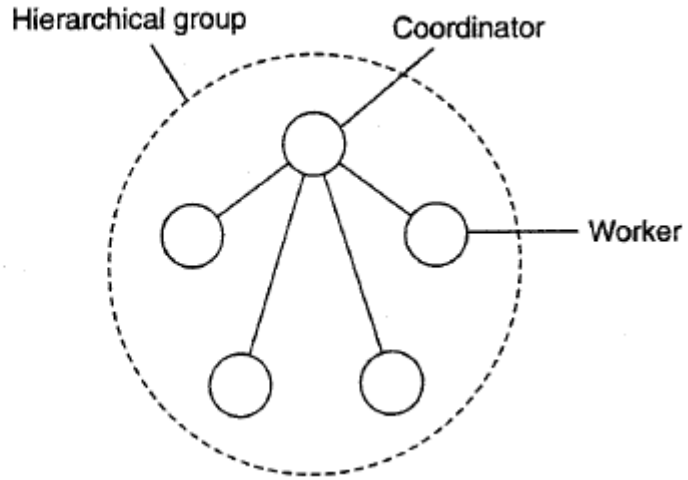
Fault tolerance

Term	Description	Example
Fault prevention	Prevent the occurrence of a fault	Don't hire sloppy programmers
Fault tolerance	Build a component such that it can mask the occurrence of a fault	Build each component by two independent programmers
Fault removal	Reduce the presence, number, or seriousness of a fault	Get rid of sloppy programmers
Fault forecasting	Estimate current presence, future incidence, and consequences of faults	Estimate how a recruiter is doing when it comes to hiring sloppy programmers

PROCESS RESILIENCE

- The key property that all group shave is that when a message
- is sent to the group itself, all members of the group receive it. In this way, if one process in a group fails, hopefully some other process can take over for it.





Failure Masking and Replication

- replicate processes and organize them into a group to replace a single (vulnerable) process with a (fault tolerant) group.
- by means of primary-based protocols
  - a group of processes is organized in a hierarchical fashion in which a primary coordinates all write operations.
  - when the primary crashes, the backups execute some election algorithm to choose a new primary.
- through replicated-write protocols.

Agreement in Faulty Systems

- Synchronous versus asynchronous systems. A system is **synchronous** if and only if the processes are known to operate in a lock-step mode.
- A system that is not synchronous is said to be **asynchronous.**
- Communication delay is bounded or not. Delay is bounded if and only if we know that every message is delivered with a globally and predetermined maximum time.

Agreement in Faulty Systems

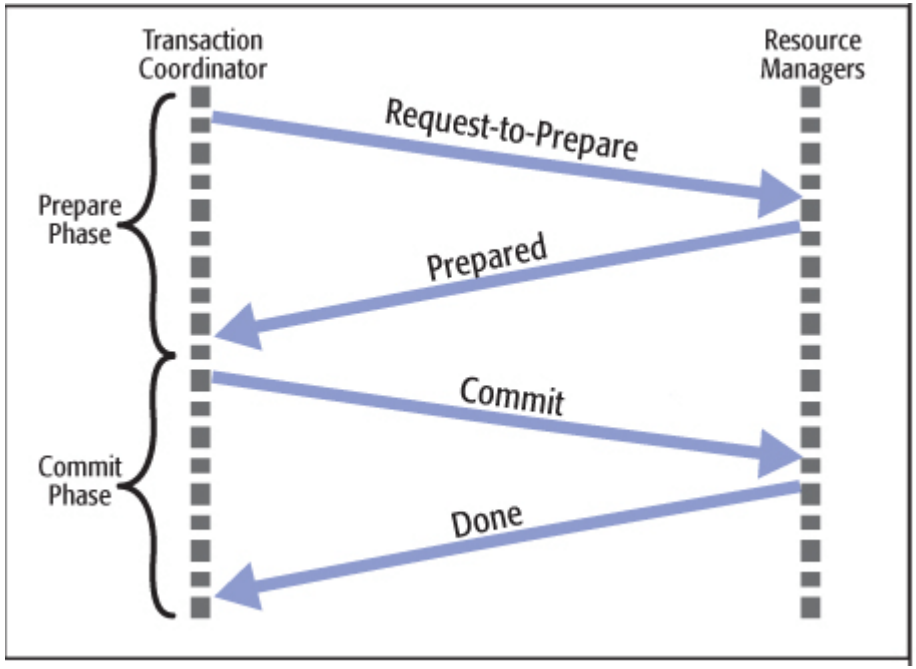
- Message delivery is ordered or not. In other words, we distinguish the situation where messages from the same sender are delivered in the order that they were sent, from the situation in which we do not have such guarantees.
- Message transmission is done through unicasting or multicasting.

Failure Detection

- **Pinging**
- gossiping -in which each node regularly announces to its neighbors that it is still up and running.
- Distinguish network failures from node failures. One way of dealing with this problem is not to let a single node decide whether one of its neighbors has crashed.
- Instead, when noticing a timeout on a ping message, a node requests other neighbors to see whether they can reach the presumed failing node.

DISTRIBUTED COMMIT

- **THE TWO PHASE COMMIT PROTOCOL**
- One of the processes is the coordinator and other processes are cohorts.



Phase 1: Coordinator

- Coordinator sends a ***Commit_Request*** message to every cohort requesting the cohorts to commit.
- The coordinator waits for the replies.

Phase 1: Cohort

- On receipt of Commit request
  - If the transaction is successful
    - Writes undo and redo log on the stable storage.
    - Sends ***Agreed*** message to the coordinator.
  - Else if transaction is unsuccessful then
    - It sends an ***ABORT*** message to the coordinator.

- **Phase 2 : Coordinator**
  - If all the cohorts reply ***agreed*** and the coordinator also agrees, then the coordinator writes a **commit** record in to the LOG.
  - Otherwise it sends an ABORT message to all the cohorts.
  - The coordinator waits for acks from each cohort.
  - If an ack does not arrive from any cohort within time out period, the coordinator resend the commit/abort message to that cohort.
  - If all the acknowledgements are received , the coordinator writes a COMPLETE record to the log.

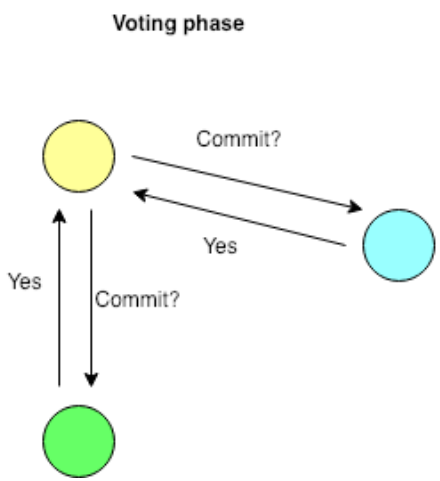
- **Phase 2 : Cohorts**
- On receiving a COMMIT message, a cohort releases all the resources and locks held by it for executing the transaction, and sends an acknowledgement.
- On receiving an ABORT message , undoes the transaction using the undo log record, releases all the resources and locks held by it for performing the transaction, and sends an acknowledgement.

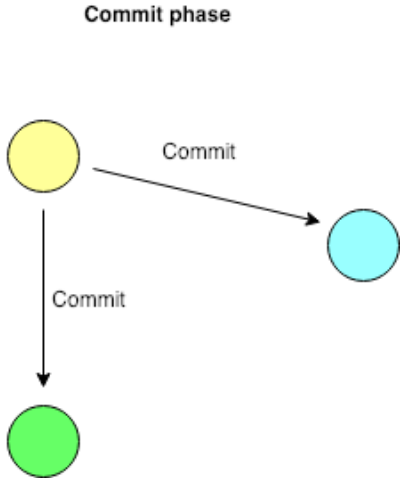
VOTING PROTOCOLS

- With the voting mechanism, each replica is assigned some number of votes, and a majority of votes must be collected from a process before it can access a replica.
- The voting mechanism is more fault-tolerant than a commit protocol in that it allows access to data under network partitions, site failures and message losses without compromising the integrity of the data.

Algorithm

- Every replica is assigned a certain number of votes`
- Every site has a **lock manager.**
- Every file has a **version number.**
- Every replica is assigned a certain number of votes.
- Read and write permitted only if a certain number of votes are obtained(read quorum) and (Write quorum) by the requesting process.





Recovery of a System

- **Forward error recovery:**
  - If the nature of errors and damages caused by faults can be completely and accurately assessed, then it is possible to remove those errors in the process's state and enable the process to move forward.
- **Backward error recovery:**
  - If it is not possible to foresee the nature of faults and to remove all the errors in the process's state, then the process state can be restored to a previous error free state of the process.
- **STATE BASED APPROACH:**
  - **Recovery Point/ Checkpoint**: In the state-based approach or recovery, the complete state of a process is saved when a recovery point is established, and recovering a process involves reinstating its saved state and resuming the execution of the process from that state.

Reference

- https://programmerprodigy.code.blog/2021/07/07/fault-tolerance-and-recovery-in-distributed-systems/
- https://www.scirp.org/html/1-9702032_61986.htm#txtF6