TT-1

Q: * Suppose, multiplayer online game streaming
— which is better? TCP or UDP?

→ $\boxed{\text{UDP}}$

Reason: TCP → connection oriented protocol
→ provides reliable, ordered and error-checked delivery of data
→ data integrity is priority
→ detect and recover error
→ ensure that all data arrives intact.

But, doesn't provide:
1) affect real-time responsiveness required for gaming [Latency]
2) more overhead due its additional features.

Pros of UDP:
1) no error checking mechanism, no overhead connection setup → faster than TCP
2) more suitable for real-time applications

* If priority is real-time interactions and can tolerate occassional packet loss → UDP

* If priority is data reliability and integrity and can tolerate slightly higher latency → TCP

* many dev use combo of both.

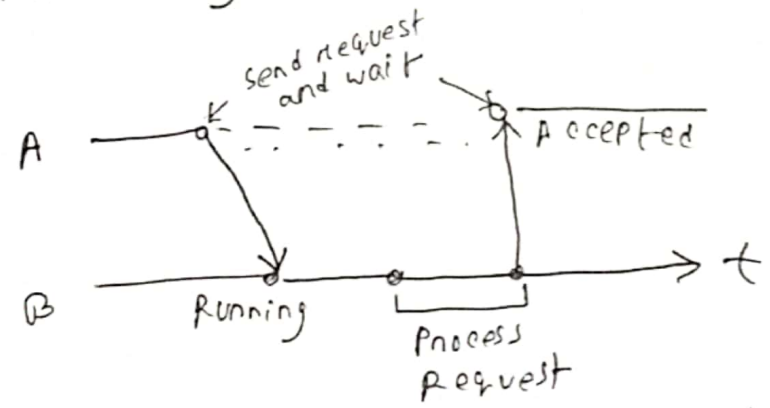
Q: What's message oriented communication? Explain:

a) Response based transient synchronous

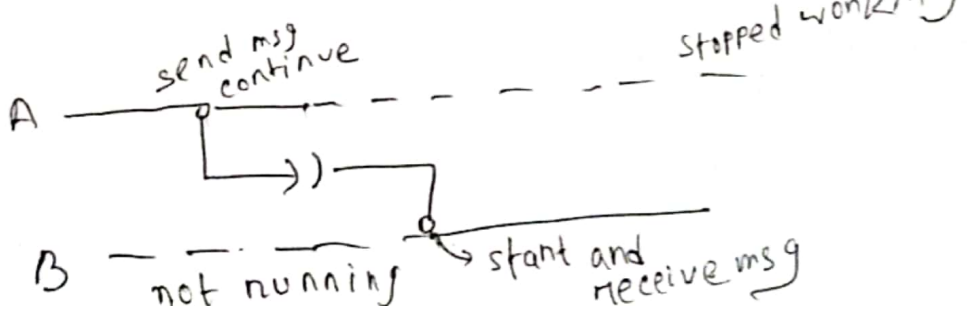
b) Persistent asynchronous

→ way of communication between sender and receiver's process, in term of sending and receiving explicit message between them.

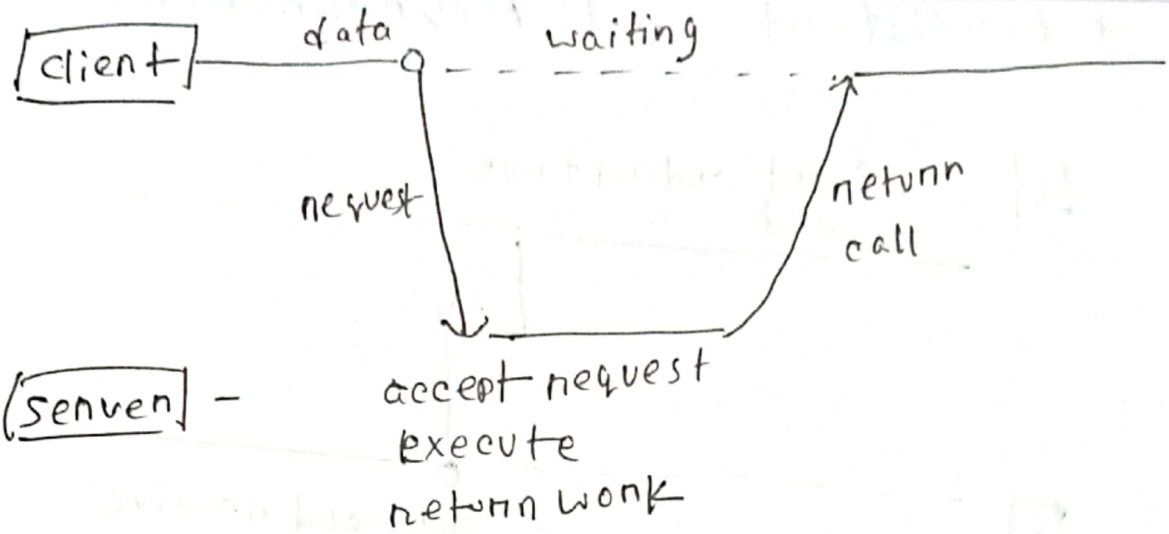
(a)



(b)



RPC:



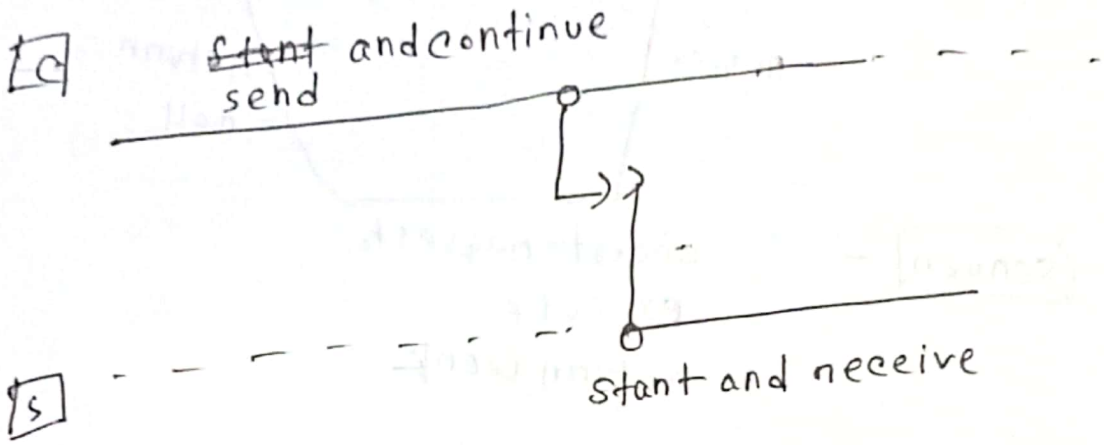
Persistent: when one of the sender/receiver is offline, message is stored
when online → find message

transient: when one offline → message disappears
both should be online

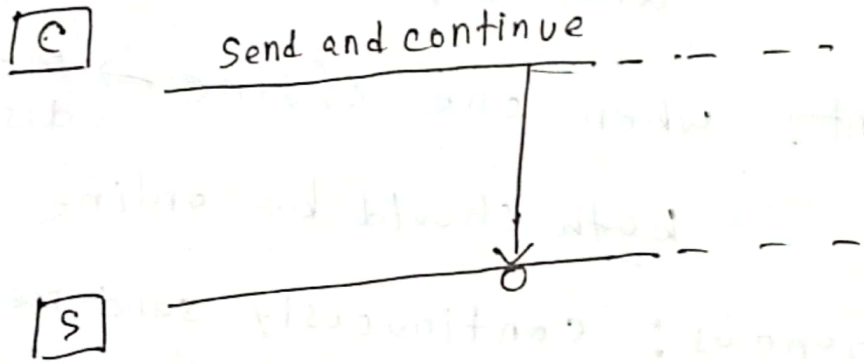
Asynchronous: continuously send message
- not wait for server to respond
- not blocking - independent

Synchronous: send message and wait
server send '~~SYN~~ ACK' to client.
- dependent

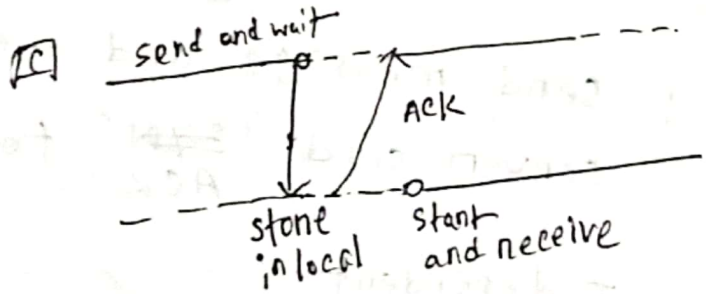
* Pensistent and Asynchnonous:



* Transitent and asynchnonous:

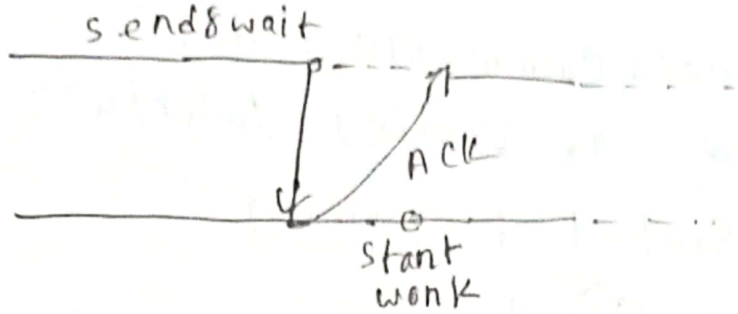


* Pensistent and synchnonous:



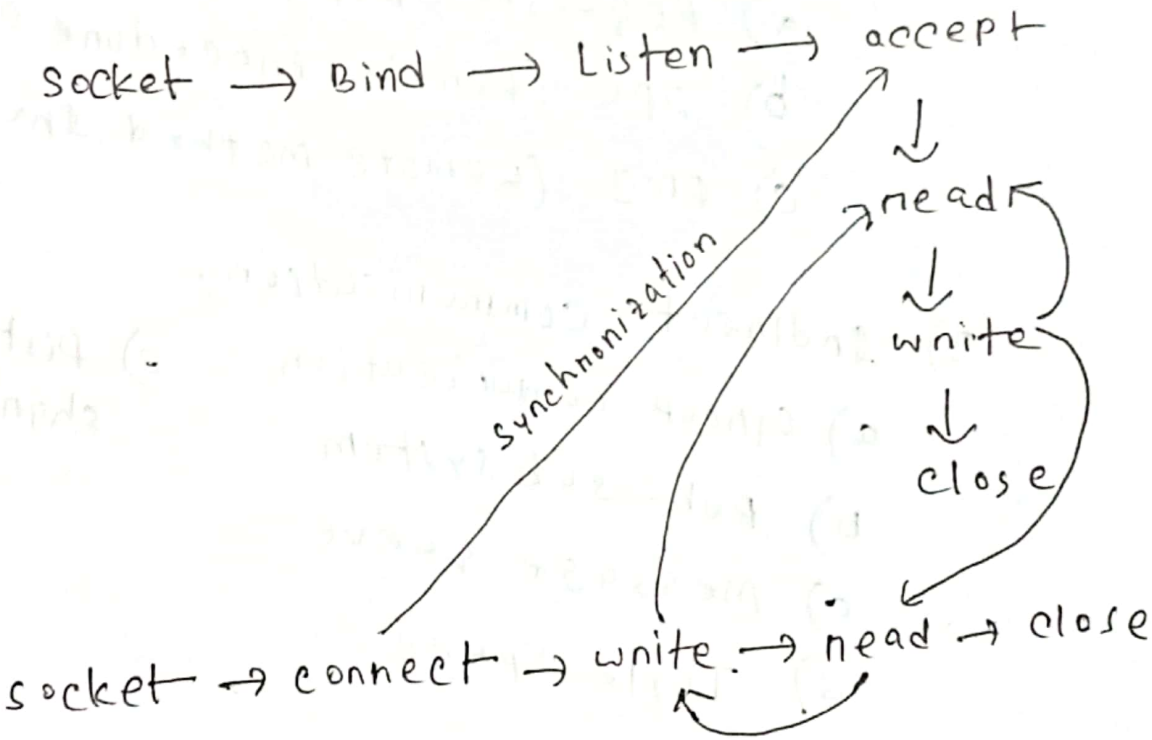
* Transistent and Synchronous:

[C]



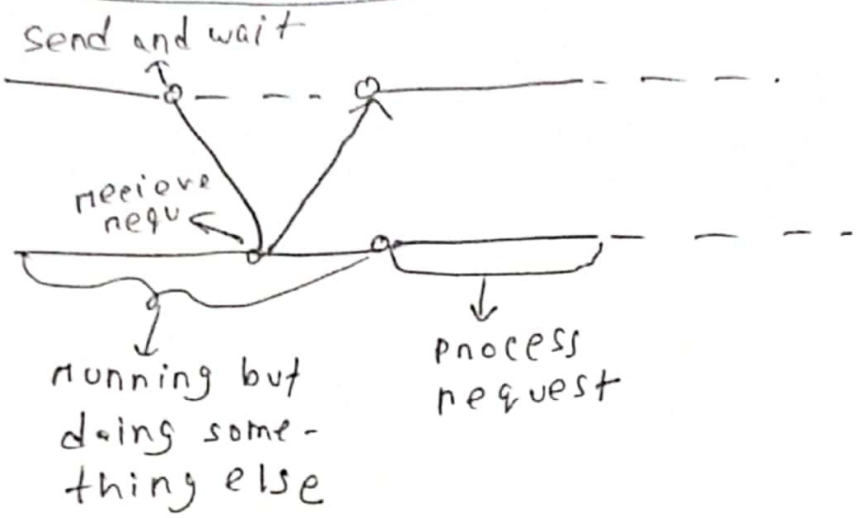
Benkley - socket:

[S]

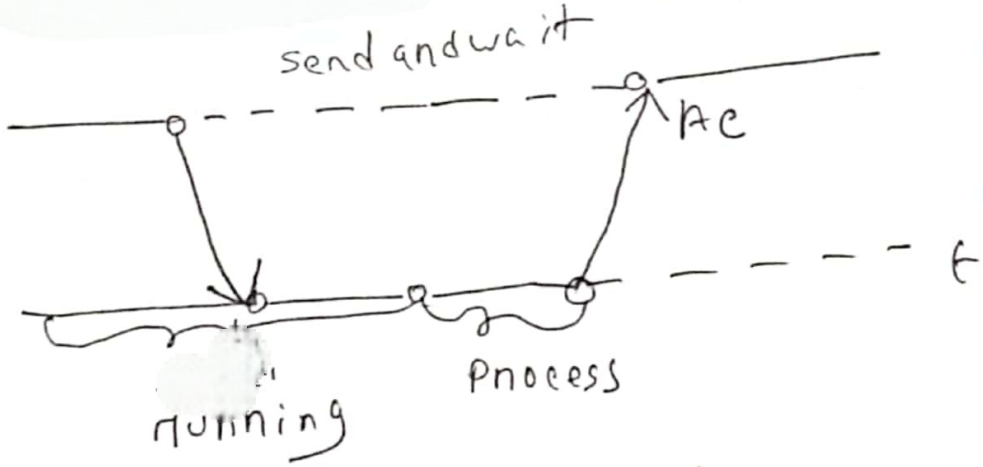


[C]

Receipt - based transient synchnomous:



Response - based:



Q: Real-time chat app using gRPC. Write a .proto file that defines message types and service methods for this app.

a) message types including sender, receiver, timestamp and message content.

b) service methods:

i) send message client to server

ii) recieve on server and broadcast to clients.

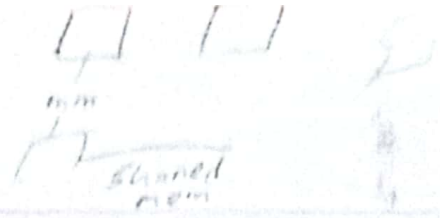
```
syntax = "proto3";
// defining message types
```

→ (a)

```
message chatMessage {
    string sender = 1;
    string receiver = 2;
    int timestamp = 3;
    string content = 4;
}
```

(b)

```
// define service
service chatservice {
    rpc sendmessage (chatMessage) returns
        (chatmessage);
rpc BroadCastmessage (chatMessage) returns
        (google.protobuf.Empty);
```



Q: What advantages of DSM over MP?

→ 1) increased speed and efficiency due to avoidance of copying and transferring data between processes

2) Direct access to data
3) Reduced communication overhead
4) Synchronization simplicity
5) Global shared memory
6) Easy programming.

Q: UDP or TCP for voice-over IP app?

→ UDP

1) Low latency - ensures immediate convo flow.
2) Low overhead
3) for voice communication, data loss is accepted
4) simple and straightforward.

Q: Dear, Bear, River, Car, Car, River, Deer, Car, Bear.

* Count the occurrences of these words using MapReduce.

→ MapReduce is a programming model and

processing technique designed for distributed computing ~~of~~ on large datasets.

2 main phases

1) <u>map phase</u>: data processed and organized into key-value pairs.

2) <u>Reduce phase</u>: results from map-phase are ~~reduced~~ gathered based on keys.

<u>Ex:</u> <u>Input</u>: D^{ean}, B, R, C, C, R, D^{een}, C, B

<u>Map Func:</u> D^{ean} - 1, B - 1, R - 1, C - 1, C - 1, R - 1, D^{een} - 1, C - 1, B - 1

<u>Shuffle and sort:</u> B - 1, B - 1, C - 1, C - 1, C - 1, D_{ean} - 1, D_{een} - 1, R - 1, R - 1

<u>Reduced phase:</u> B - 2, C - 3, Dean - 1, Deen - 1, R - 2

```
//map phase
function map (key, value) {
    for each word in value.split()
        emit (word, 1)
// reduce phase:
func reduce (key, value list) {
    emit (key, sum (value))
```

code:

B+ Tree:

order $m = 4$

max child $= m$ max key $= m-1$

min child $= \lceil \frac{m}{2} \rceil$ min key $= \lceil \frac{m}{2} \rceil - 1$

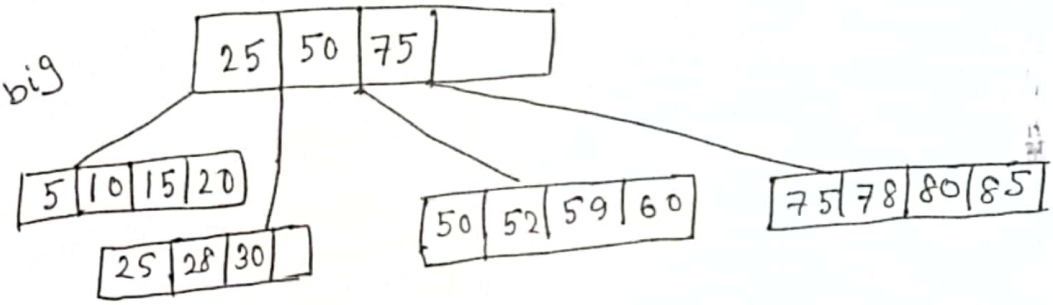
* only stored in leaf nodes
* property of Binary Search Tree

Ex: 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

→ max key $= 5 - 1 = 4$ min $= 2$

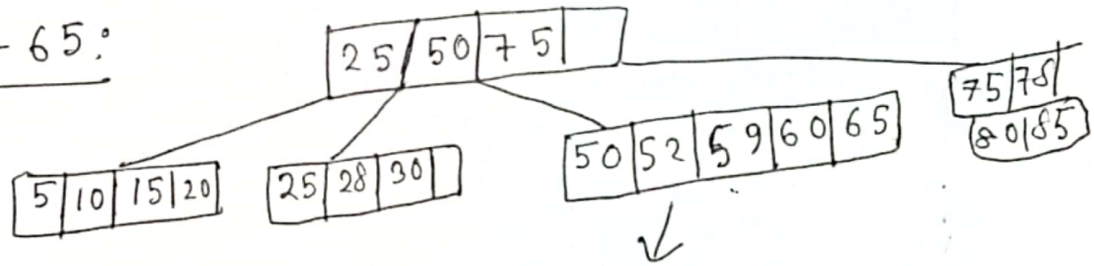
max C $= 5$ min C $= \lceil \frac{5}{2} \rceil = 3$

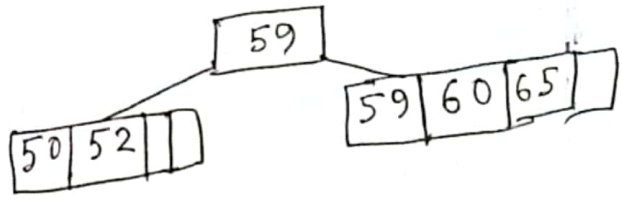
to left: small
to right: equal or big



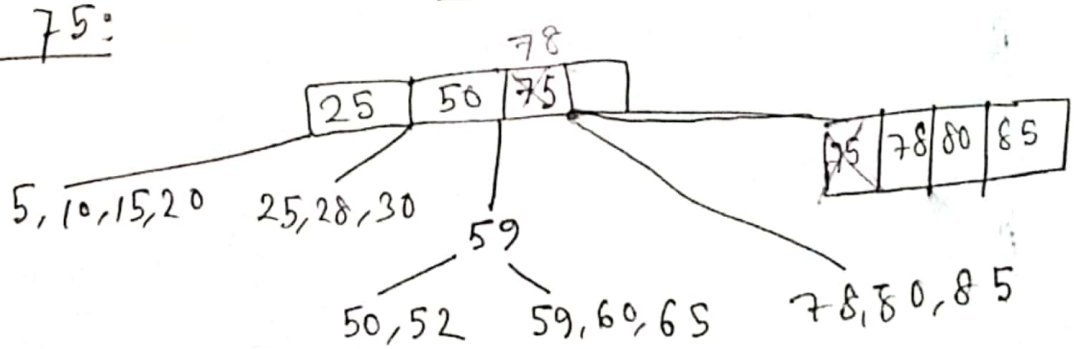
Q:

Insert 65:





Delete 75:

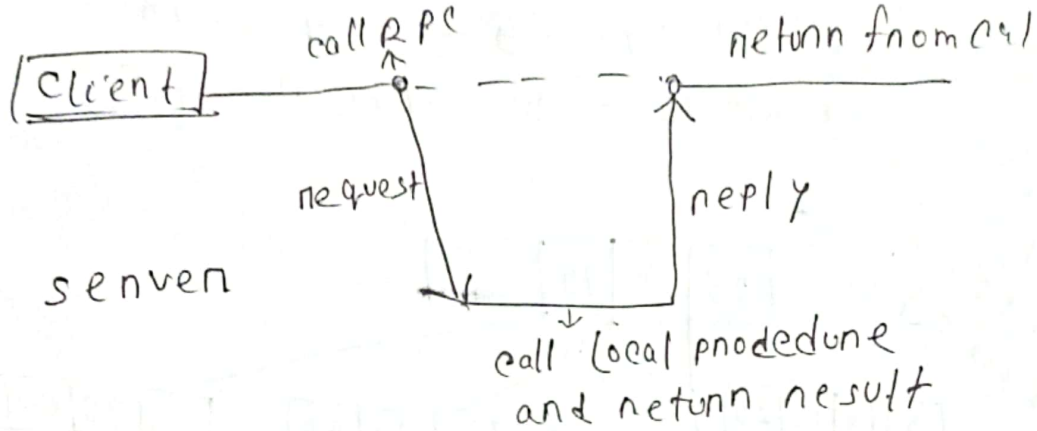


RPC – Remote Procedure Call

RMI – Remote method Invocation

What is RPC and RMI?

→ RPC executes a procedure at remote site and ship the result back.



→ RPC is a protocol that allows a program execute procedure of another program locate in other device without developer explicitly coding details for remote interaction.

→ used for procedure based app.

RMI: used for distributed object system

→ object oriented

→ java-based system that allows an object running in JVM to call methods on an object running in ~~JV~~ another JVM.

Q: Stub in RPC:

client-side proxy that acts as a representative of a remote object. It allows the client to call methods on the remote object as if it were a local object.

→ responsible for:

1) packaging parameters
2) making remote call
3) returning results.

Q: Thrashing in DSM: refers to excessive page swapping between local and remote memories due to high contention for shared data.

→ cause performance degradation; system spend time in transferring data between nodes rather than performing useful computation.

Q: Marshalling and unmarshalling?

→ Marshalling:

Data of a program
↓ convert
Transmittable format
↓
Reconstructed into original form

Unmarshalling: (reverse)

transmitted/stored format
↓
original form of data

Q: Election service: which is recommended?

→ at-least-once call semantics

→ at-most-once

Ans: At least once: ensures that every user request to cast a vote will be processed at least once by the election service.

- Pro: No valid votes - lost even if network failure

Cons: puplicate vote may happen

At most once: each user request to cast a vote is processed at most once

Pro: No duplicate votes

Cons: Some valid votes lost

* If data accuracy (valid vote) is priority → at least once

* If duplicate is unacceptable → at most once

* Idempotent operate:
  - produce same result no matter how many times they're called
  - avoid duplicate - no loss of valid vote

Q: Heterogeneity: mixture of diff. h/w, s/w, network protocols, OS, data formats, language, framework etc.

Q: Serialization: process of converting ds or objects into a format that can be easily stored, transmitted or reconstructed.

→ used for data exchange between different system or saving state of an object

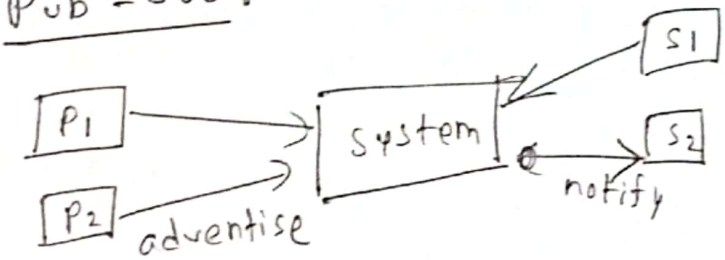
Q: Scenenio of multicast communication:

→ used in a live streaming application whene a single stream of data needs to be sent to multiple recipients simultaneously. Instead of sending individual copies to each recipient, it allows for efficient 1 to many communication.

Q: Space uncoupling in DSM

→ refers to the ability of DSM to allow components to operate independently in their local address space. This enables flexibility and scalability.

Q: Pub - Sub: asynchronous, heterogeneous.



Ex: ① Real-time stock market update

② Social media notify about update

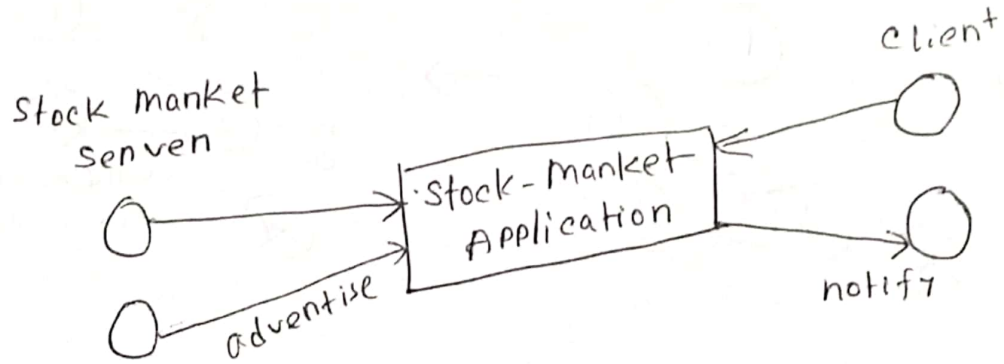
③ youtube

Q: * Application of Distributed System in Web.

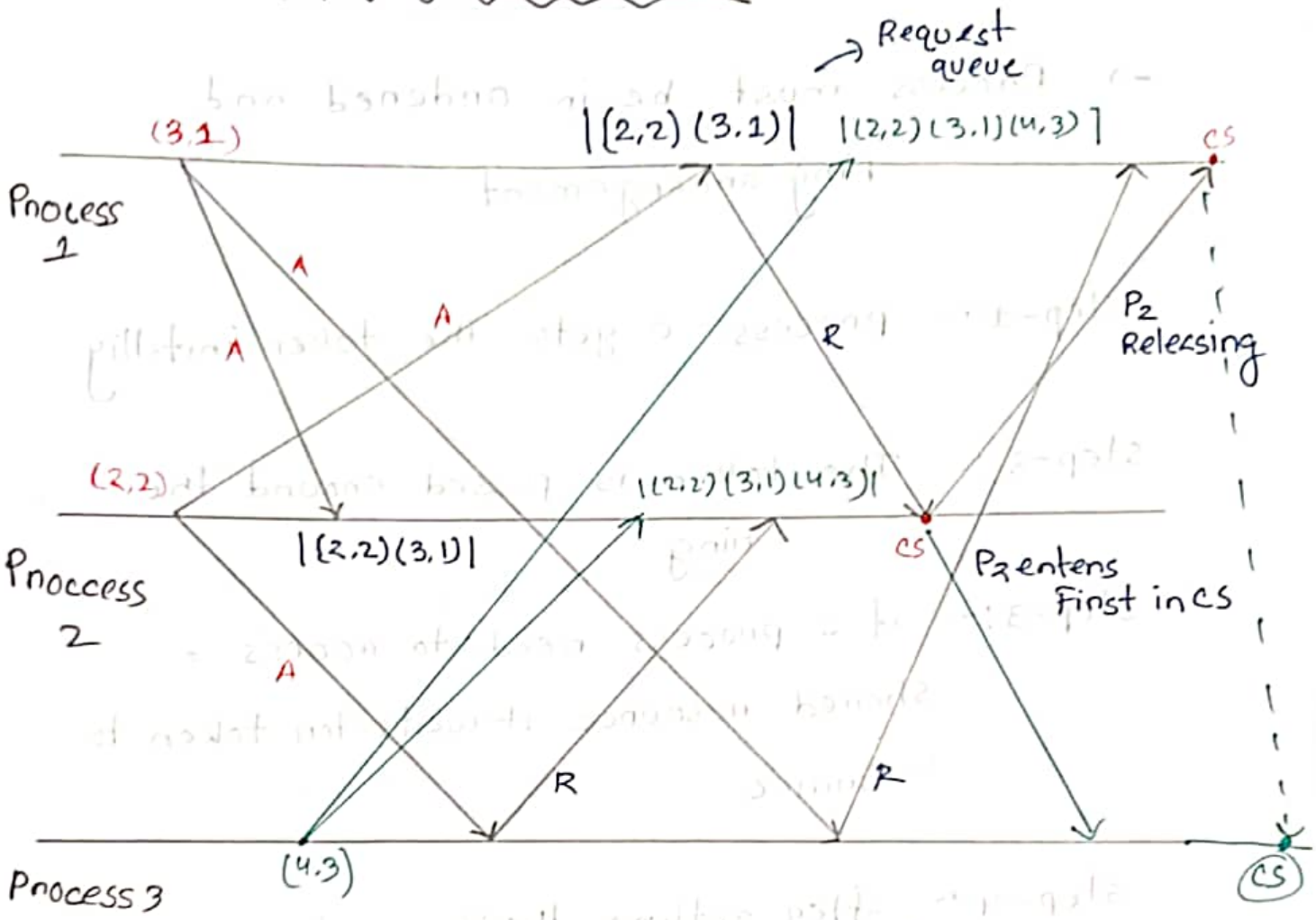
→ 1) Load Balancing
2) Content Delivery Networks
3) Distributed DB
4) microservice archi
9) scalability
5) caching system
6) Distributed File System
7) Replication for fault tolerance
8) Asynchronous messaging
10) Efficient resource utilization

Q: * 3 communication paradigm is distributed system:
1) Inter-process communication
2) Remote invocation
3) Indirect communication.

Q:



Lamport total ordering



R = Reply message

A = Request message

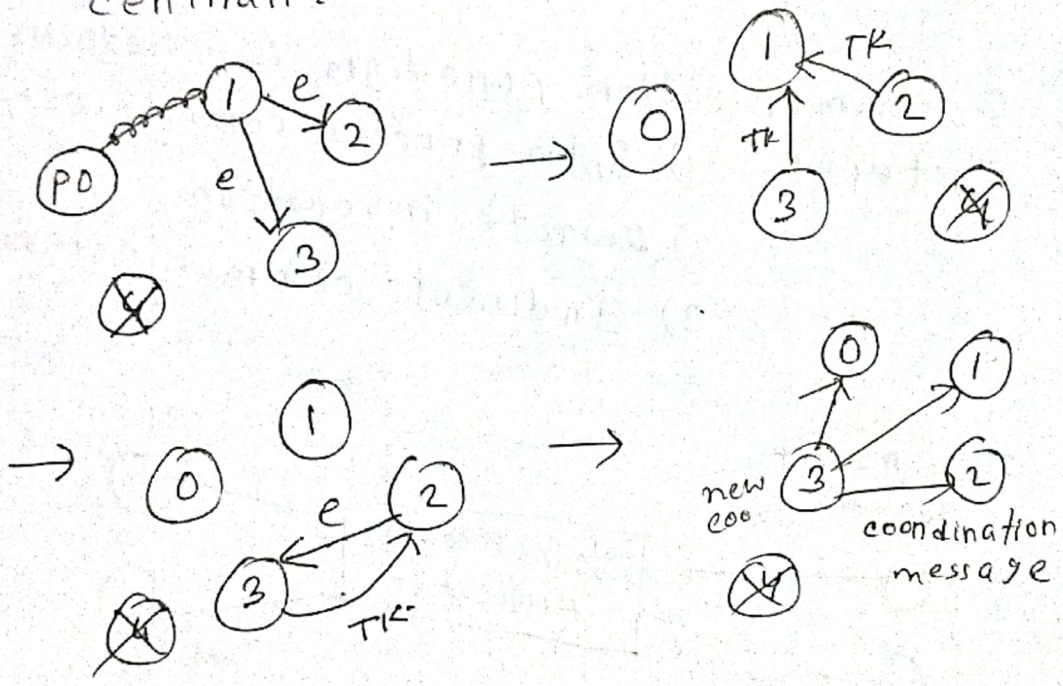
Sequence entering :-

P2, P1, P3

Q: Why is election process required in distributed system?

5 process : P0 to P4, P1 detects that P4 dead, explain new election process using bully algorithm.

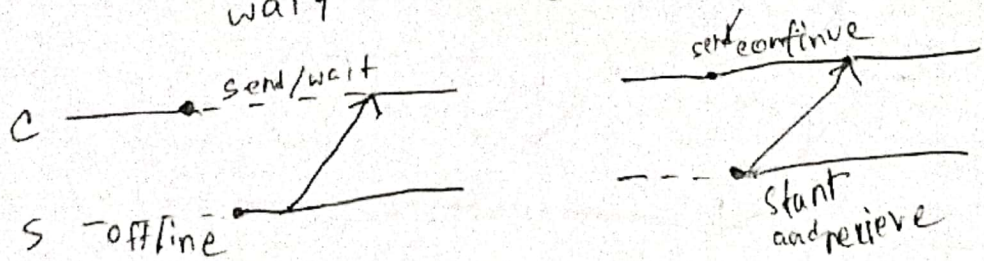
→ select designated coordinator and make centralized decisions.



Q: Sychnonous / Asynchnonous RPC.

↓ Send and wait

↓ send and continue



→ WWW properties same

Q: Common Properties of distributed system:

1) Concurrency
2) Lack of Global Clock
3) Fault tolerance
4) Communication and message passing
5) Heterogeneity
6) Scalability
7) Decentralization
8) Transparency
9) Consistency and replication
10) Security
11) Resource sharing
12) Transparency

Q: SUST (monolithic or microservice?)

→ Monolithic app is a type of sw archi where all the components and functionalities of app are tightly integrated into a single codebase and executed as a single unit.

→ microservice can scale individual services independently.

Reasons: 1) Scalable 2) Flexible 3) Isolated service 4) Technology diversity 5) Ease of maintenance 6) Independent deployment

Communication Entities: UI service, authentication, course mgmt, faculty, library, billing and finance.

communication and notification service.

2 mechanisms: 1) RESTful APIs
2) message queues

Q: Simple e-commence website:

* Communication entities:
1) UI 2) server-side app 3) DB
4) payment gateway 5) inventory mgmt system

* mechanisms: 1) RESTful APIs
2) message queues

Q: WWW is a distributed system?

Q:

Thread	Process
1) smallest unit of execution within a process, share the same memory and resource	1) independent program, own memory and resource, isolated from each other
2) Low resource overhead	2) High overhead
3) Direct communication within same process	3) IPC (MP, shared file, socket)

Thread	Process
4) Faster	4) Time consuming
5) Depends on other threads within same process	5) Independent
6) Ex: multiple threads within a web browser process for handling UI, network requests, and background tasks	6) ~~parallel~~ Ex: multiple instance of text editor

Q: Relationship between microservices and distributed system:

→ 1) Microservices as a distributed system: Micro-service architecture is a specific implementation of a distributed system. In a microservice archi, the individual services act as distributed components that communicate over a network.

2) Distributed Nature of Microservices: Micro-services inherently exhibit the characteristics of distributed systems. They communicate over a network, can be deployed on different servers, and may use diverse technologies.

3) Autonomous communication: microservices communicate with each other through well-defined APIs, and this communication is typically based on principles of distributed system such as message passing, RESTful APIs.

4) Scalability and fault tolenance:

Q: Mobile code: ability for running program, codes on objects to be moved from one machine/app to another.

→ transmitted from a host to be executed on a client machine, without client's knowledge.

→ Ex: Java, Javascript, Postscript, PDF

Q: What is MPI? Based on type of send, define diffenent communication mode?

→ MPI (message passing Intenface) is a standardized and pontable message passing system designed fon panallel computing.

→ provides a set of functions and conventions fon sending and receiving messages between processes in panallel application.

→ 4 modes of communication:

1) Synchnonous: send and wait fon receiven to nesponse. Till then block.

2) Asynchnonous: send and continue.

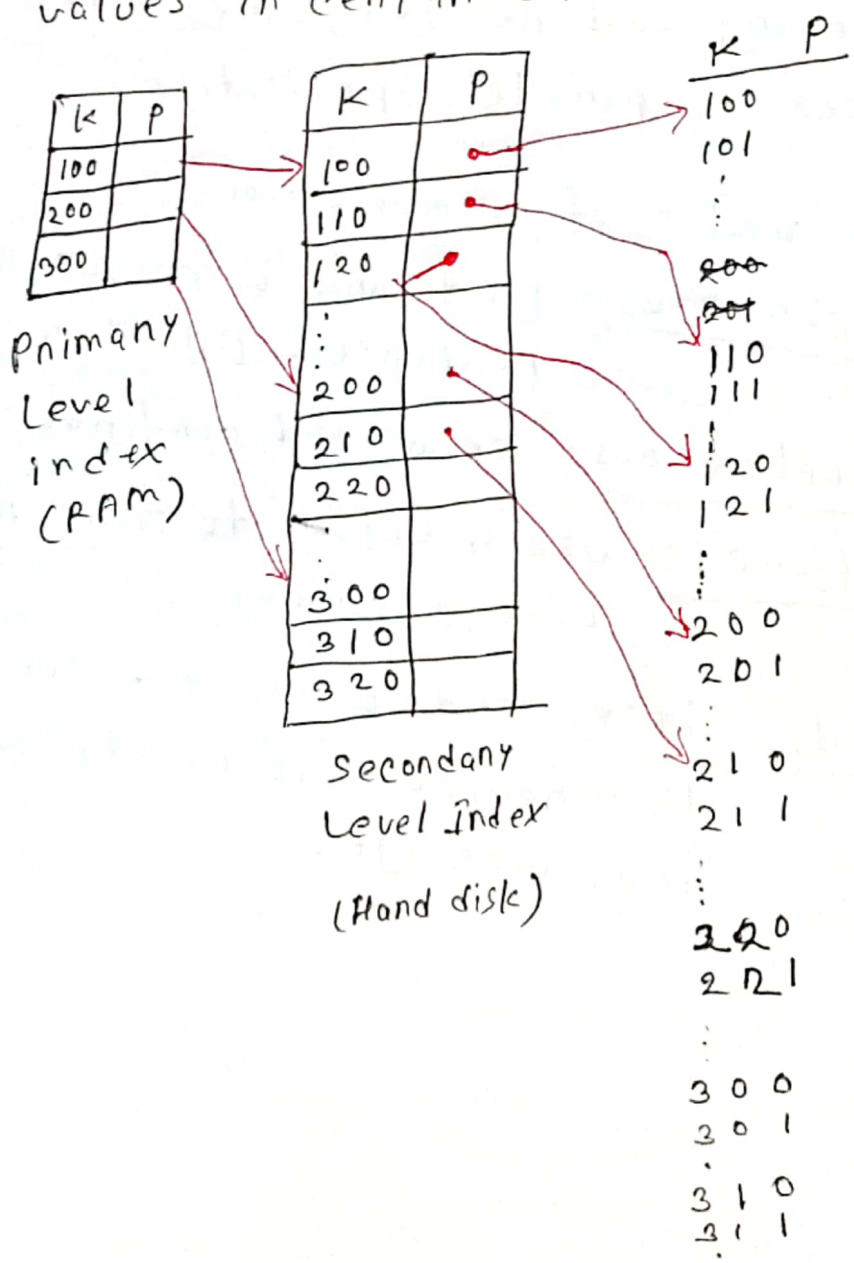
3) Buffened: use a buffen to stone message befone sending.

4) Ready: it's used when the senden ensunes that necieven is ready to necieve the message.

Indexing

Q: What is indexing?

→ a DS technique that improves the speed of data retrieval operations on a DB table. It involves creating an index, which is DS that provides a fast and efficient way to look up rows in a table based on the values in certain columns



Primary — sparse, dense

Secondary	multilevel	Hashing — Static, Dynamic
Clustering	B-Tree	ordered

Q: 2 levels of indexing mechanism:

① B tree Index: balanced tree DS commonly used for indexing in DS. It allows for efficient search, insertion and deletion, easy maintain

② Hash index: uses a hash function to map keys to index location.

Effective → equality search

not " → range query / ordered search

Q: Secondary Index: is created on a non-primary key column.

secondary index

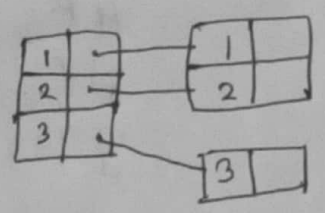
ID	Name	Dept	Salary
101	A	HR	5k
102	B	IT	6k
103	C	HR	5.5k

Dept	ID
HR	101
HR	103
IT	102

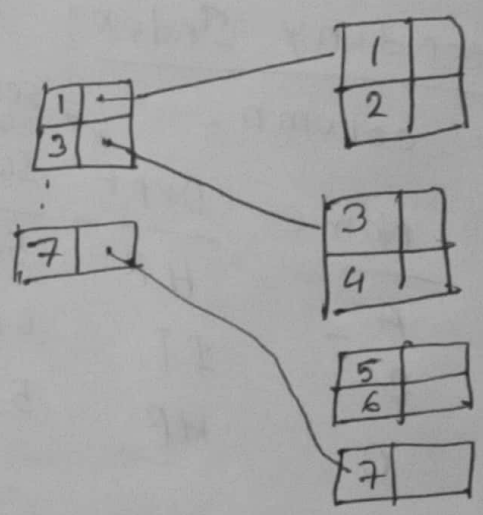
* Suppose we want to fetch all employees who belong to the "IT" dep. Instead of scanning the entire table, the secondary index allows us to quickly identify relevant ID values associated with the "IT" dept. This speeds up retrieval operations based on the secondary index column.

Q:

Dense	Sparse
→ contains an entry for every search key value in data file	contains entry only for some of the search key value in data file
→ larger	→ smaller
→ time to locate data: fastest	→ slower
→ more overhead	→ less overhead
→ time for computing in RAM less	→ more

→ 

Ex:



Q: Advantages of Sparse Index over dense index:

→ ① space efficient

② Less overhead

Q:

Ordered	Sparse
- maintains sorted order of key values.	- may not maintain a strict order of key values
- suitable for range queries	- not suitable for range queries

Q: Cons of sparse Index:

1) Slower range queries
2) More time needed for search
3) More time for RAM computing.

Q:

S3

T_1	T_2
~~R(A)~~	
~~W(A)~~	
	~~R(A)~~
	~~W(A)~~
~~R(B)~~	
~~W(B)~~	
	R(B)
	W(B)

S6

T_1	T_2
~~R(A)~~	
~~W(A)~~	
~~R(B)~~	
~~W(B)~~	
	R(A)
	W(A)
	R(B)
	W(B)

which one is conflict serializable? (both)

→ S3:

$T_1 \rightarrow T_2$

conflict serializable

S6:

$T_1 \rightarrow T_2$

conflict serializable

Q: Why non-recoverable schedule.

T_1	T_2
R(A)	
W(A)	
	R(A)
	commit

if error roll back ← R(B)

T_2 already commit, can't revert changes made by T_2 → inconsistency

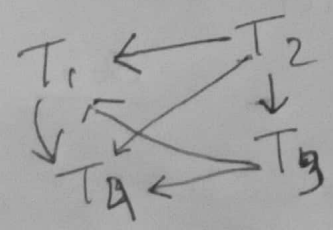
Q: Serialization: refers to the process of ensuring that a series of transactions are executed in such a way that the final result is equavalent to some serial execution of the transactions.

Q:

$T_2 \rightarrow T_3$	$T_3 \rightarrow T_4$
$T_2 \rightarrow T_1$	$T_1 \rightarrow T_4$
$T_3 \rightarrow T_1$	$T_2 \rightarrow T_4$

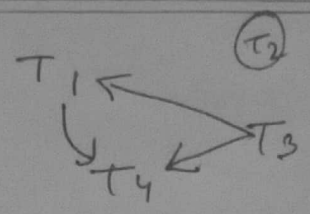
→ determine conflict seriaziable/recover on not.

Ans:



no loop → CS

$\boxed{T_2}$ indegree 0 :
remove



Now, $\boxed{T_3}$ 0 :
remove:

T_1
$\hookrightarrow T_4$

$\rightarrow$ $\underline{T_1 \text{ remove:}}$

T_4

Ans:

$\boxed{T_2 \rightarrow T_3 \rightarrow T_1 \rightarrow T_4}$

$\rightarrow$ recoverable