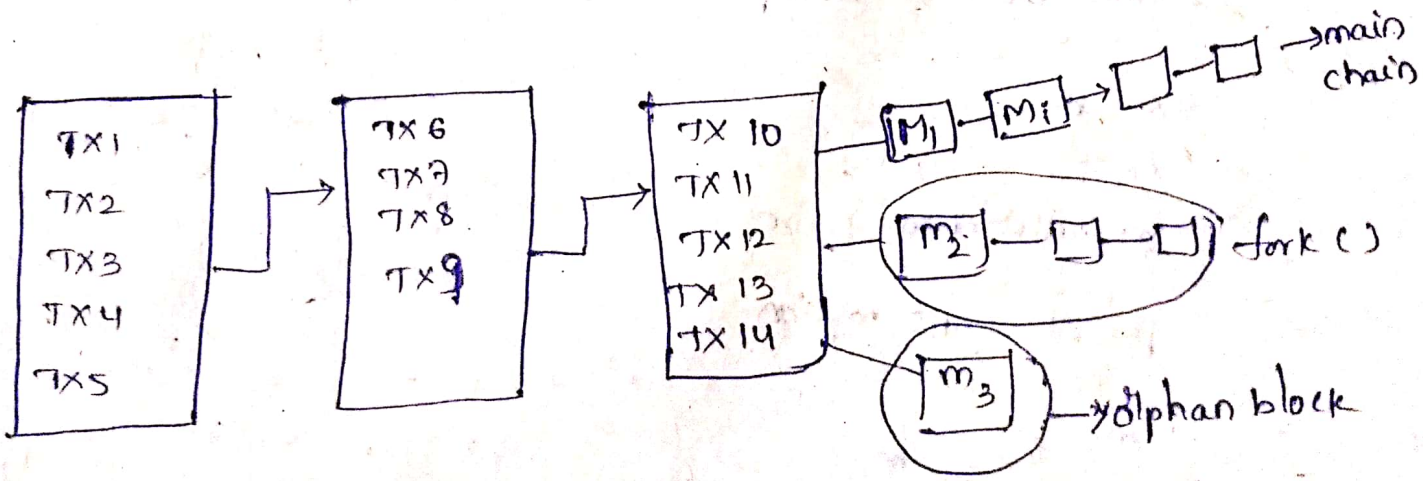
51% Rule → The block which is created newly by other miners. A node in n/w whether to accept & reject (drop) the node will depend on the 51% Rule.

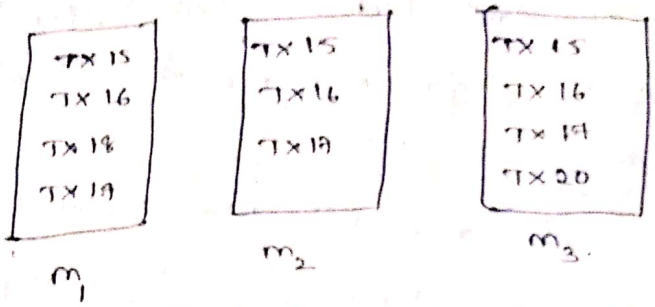
ie. if a node(n_i) which received the newly created block more than half of the nodes in the n/w, then it will accept & add in the current chain of his own copy of block chain. again broadcast to its peers remaining will validate them. so on.

BitCoin Consensus (Rules) Agreement :-

- When ever we get a new transaction, the miner will take the transaction that are not in the previous block chain. ie, they should take new transactions only. (which are not in BC)



fork = chain of orphan block



Consensus Mechanism used in block chain

- Proof of work: (H_k)

H_k is the proof that people have done some work

H_k = Hash (H_{k-1} || merkle root || nonce)

26/08/19

Application specific integrated chips (ASIC) miners.

Disadvantages of proof of work:

- If 1000 miners are writing some block but only 1 miner block is accepted and other 999 miners blast is wasted.
- computational power
- Power consumption
- Time

Stake: miners with 1% of bitcoins can mine 1% of blocks

- Proof of Stake:
  - If miners are allowed to mine only upto the % of bit coins only.

Disadvantage:

- The miners who have more bitcoins, they are only allowed to mine the blocks.

[monopoly]

③ Proof of Burn:

→ transferring bitcoins to unspendable sender showing it as a proof of work.

→ here other miners see it & doesn't waste resources keep Edm.

disadvantage:

→ The miner who having more bits, will do transactions again (monopoly)

④ Proof of Elapsed time:

- Randomly ∀ miners (nodes) get some time when time is elapsed then only it will get start mining

disadvantage:

- nodes are being silent, wastage of time,

* Attacks on bitcoin n/w

- Double spending sol) validate the transaction based on the other chain in n/w

- Sybil attack [If a rich person can hire nodes > 51% nodes, than he have the control of the n/w. & can add fake nodes]

Assumption: Bit Coin n/w assume that the Richers are going to hire from same locality.

a · b · (c · d)

↓

allows only 1 node.

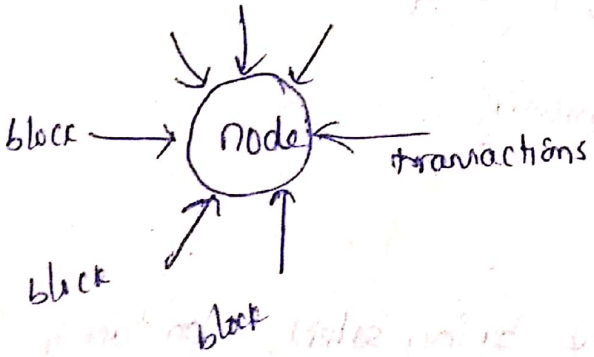
- if 2nd node comes it doesn't accept.

∴ But,

- But, The richer may hire from different locality

- Denial of Service

Sending a lots of transactions, to any node, then the node get overloaded, and spent time in processing and doesn't mine. so, we were blocking a node from mining



Sol^n Removing such a node from Block chain. by consensus mechanism. (51%) of votes by the other nodes.

27/08/19

→ 2 weeks → 2016 blocks should be generated.

2 weeks = $14 \times 24 \times 60$ ⇒ $\frac{20160 \text{ mins}}{10 \text{ mins}}$ → 2016 blocks

bitcoin n/w is designed ⇒ if < 2 weeks found 2016.

if > 2 weeks to mine 2016 blocks

difficulty level ↑ed & ↓ed respectively.

- Difficulty of mining:

$$\text{difficulty (next)} = \text{current difficulty} * \frac{\text{2 weeks in ms}}{\text{time taken to mine 2016 blocks (in ms)}}$$

Permissioned Blockchain:-

- only authenticated persones only can be in the n/w.
- It is used in mostly industries, Academics, etc

Eg. Blockchain 2.0

- Here we don't know the people's behaviour ie., malicious & not. so here also consensus mechanism. will be follow-ed

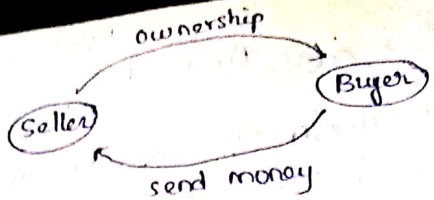
Smart Contracts

28/08/19

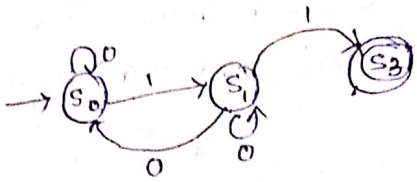
- Self executing code

an automatic code will do the work ie.,

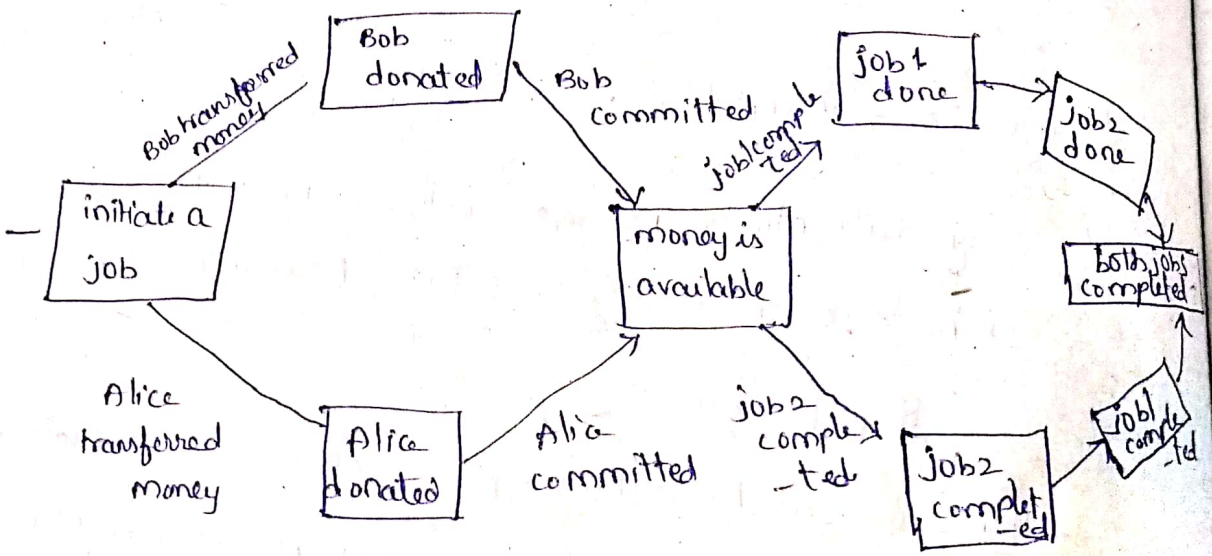
if buyer transfer money, then the owner ship is transffered to the buyer will automatically.



State machine



Crowd-funding:



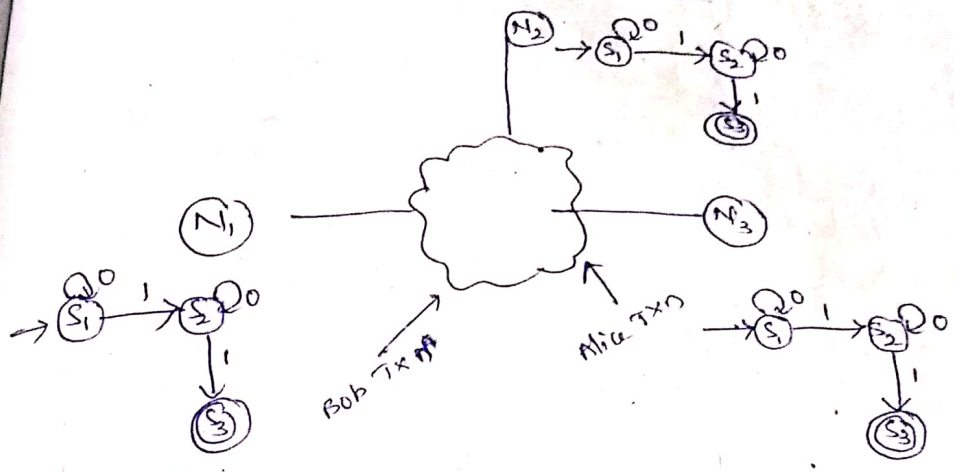
This is the way one agreement is converted to state machine

- This statemache is available at $\forall$ node
- So, we make sure that $\forall$ node will at the same state. ie,

@1) see

Synchronization b/n nodes (States of nodes).

Distributed state machine Replication:



Though Bob transaction is nearer to N_1 and 2nd transaction Alia is transferring money so, as it is nearer to node N_3. These N_1, N_2 will see them and wait for a some time and send these transactions to each node in n/w.

based on time stamp, they sort all transactions, so, that all nodes have same state of state machine.

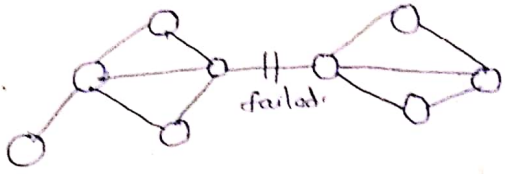
- Permisoned n/w - is a closed n/w.

  So, Smart contract is a good / available / used in closed n/w like Permissioned blockchain.

- If in case it is implemented in the permissionless blockchain then, any node may enter into n/w at any point of time, and they may not understand the contract. so, it is better to implement in the Permissioned n/w.

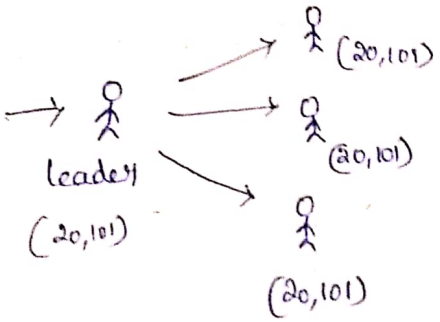
outline → 3 Types faults may happen in distributed n/w

- crash fault [when every a node crashes/ not providing service]
- network fault [n/w fails, unable to communicate with other nodes].



- Bizantine fault (malicious)

[node has becoming malicious]



(i) when a leader get crashed then followers wait for a time, if they didn't get any heart beat message, wait for some time, and one of the follower become a leader, & later leader alive, it will become follower.

(ii) n/w fault:

We have to make sure that no n/w fault and if any try to make it correct

(iii) Byzantine f

much p

problem

→ RAFT

It c

if f

a

So, R

distr

→ Log of

End

Term

(iii) Byzantine fault:

When the follower is Byzantine there may not be much problem. If a leader is a Byzantine, it will be a problem.

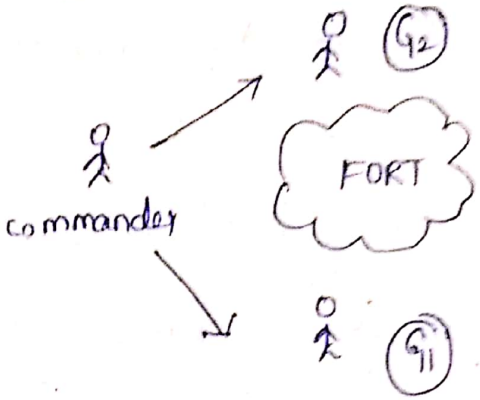
→ RAFT consensus doesn't deal with Byzantine node, It can't able to check whether a node is Byzantine or not if find byzantine it can't handle it. because it is not a fault tolerant system.

So, RAFT consensus mechanism is not used in real time distributed system.

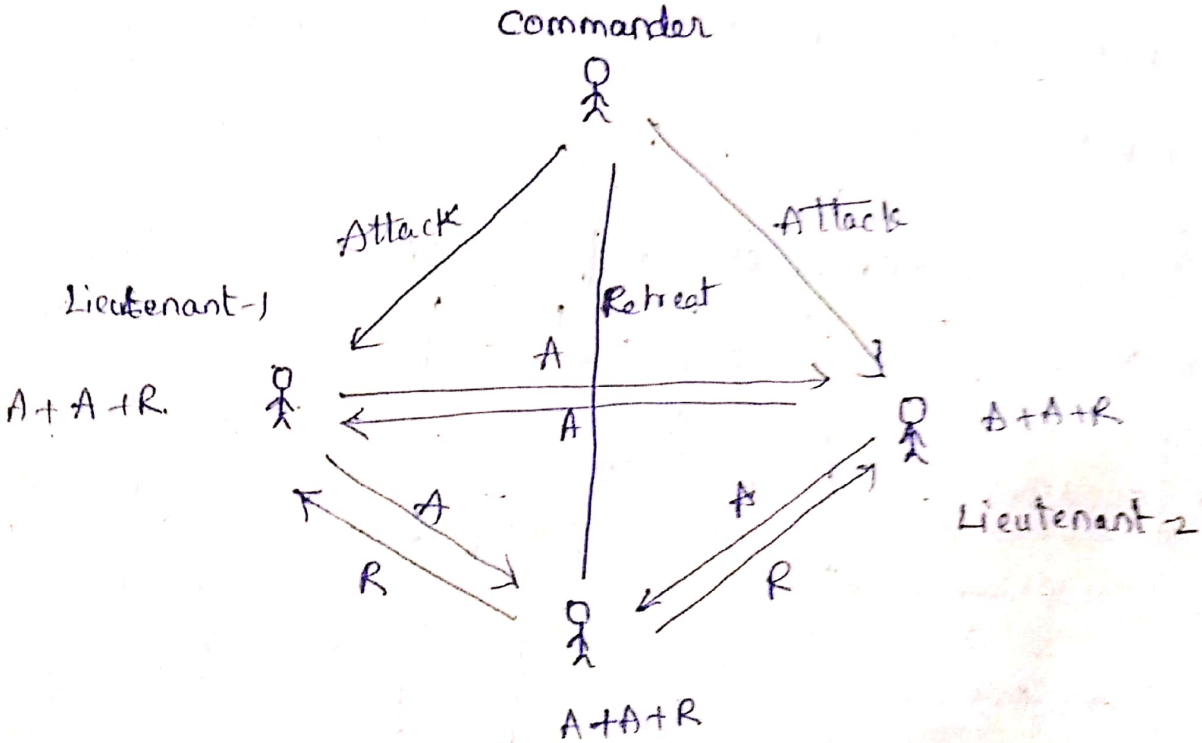
→ Log of each candidate (status of candidate)

Index		1	2	3	4	5	6
Term	candidate 1	1	1	2	2	2	2
	candidate 2	1	1	2	3	3	
	candidate 3	1	1	2	3	3	3

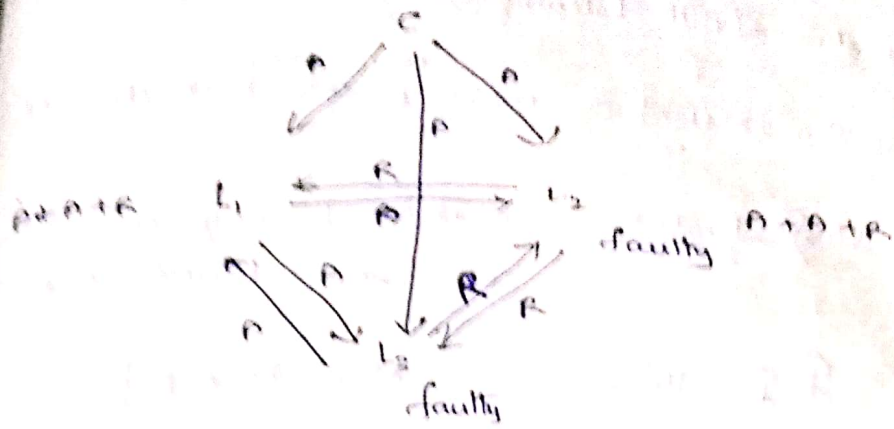
Byzantine General problem:

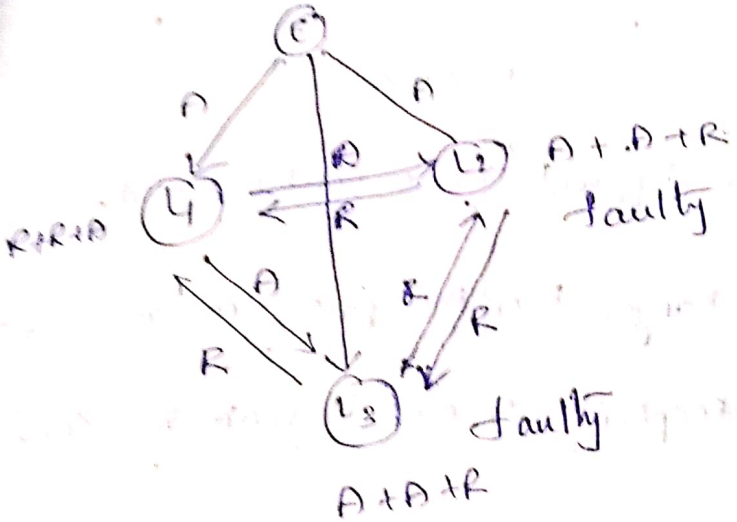


6/09/19



All are going to attack.





if even no. of nodes/ lieutenants → then it is not possible to come with correct consensus.

So if 'n' lieutenants are faulty then there must be (n+1) nodes to get correct decision.

So, total nodes = n+(n+1)

⇒ 2n+1

So, by using (2n+1) nodes, almost one can handle 'n' nodes/ n-faulty nodes.

(2n+1) nodes is excluding commander

including commander (2n+2) (including commander)

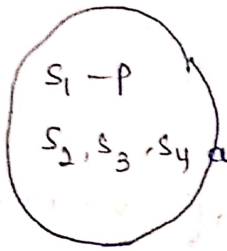
Practical Byzantine fault tolerant system.

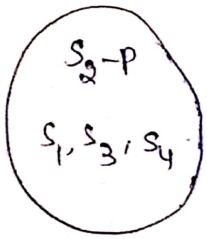
— This is algorithm is used in mostly Permissioned block chain

— terminology

 leaders → called as primary replica / called as primary node

 followers → called as backups / secondary replica





based on the primary replica different views can be formed.

So, if any primary replica changes, then the past commands passed by him/her shouldn't be accepted. In order to make the others to not accept the commands by him/her,

choose some other primary replica and change the view.

- if 'n' nodes are there, there then 'n' views are possible at max.
- ∵ primary replica may also become faulty, it should be changed so, need for changing the primary replica.

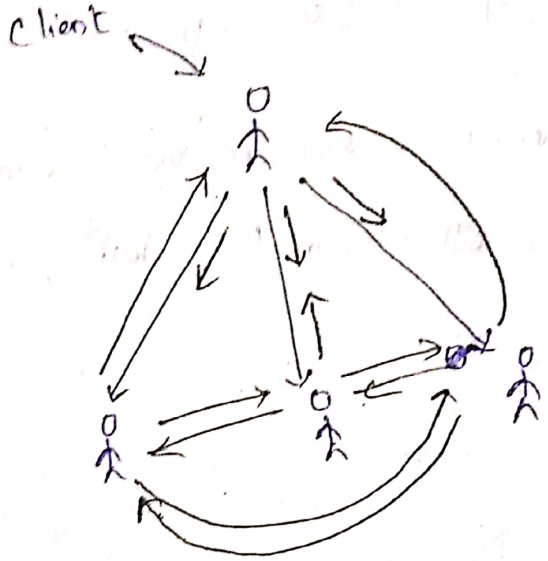
q(a)2

practical Byzantine fault tolerant system.

- $3f+1$ nodes
- f — faulty

$2f+1$ → good nodes

from these $2f+1$ → $f+1$ majority will be considered



phases:

- Preprepare phase
- prepare phase
- commit phase

↓

before preprepare phase, client will send a message to primary replica.

→ preprepare phase: primary replica will assign a seq.no 'n'.

$\langle\langle PRE\text{-}PREPARE, v, n, d \rangle_{\sigma_p}, m \rangle$

σ_p = private key of primary replica

m = actual message

v = view

n = seq.no

d = message digest

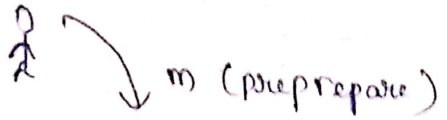
→ secondary replica will check the authenticity (&) signature of the primary replica & not

- In the same view & not, seq no is correct & not (which is expected & not).

Prepare phase: <PREPARE, v, n, d, i>
σ_i

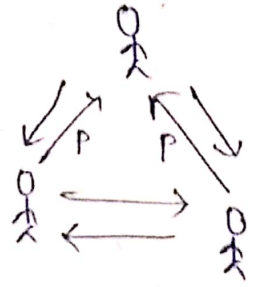
i - identity of node (secondary node).

In order to know from which node message is coming from, all secondary nodes will include their identity and also with digital signature.



m should be = p.

ie, ∀ node will check the prepare msg



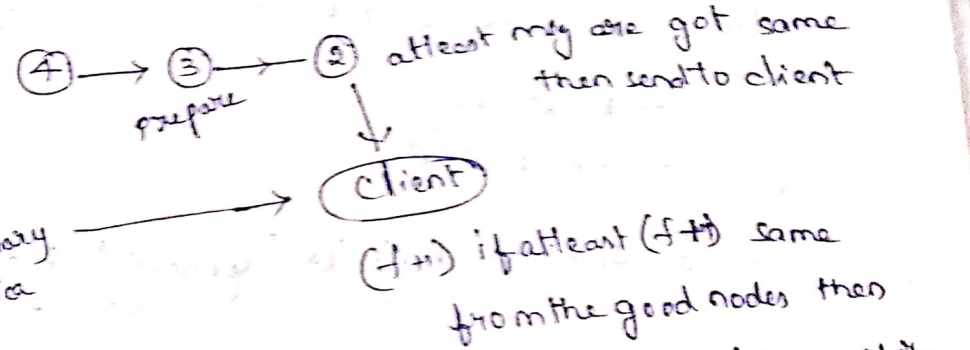
P = Prepare msg.

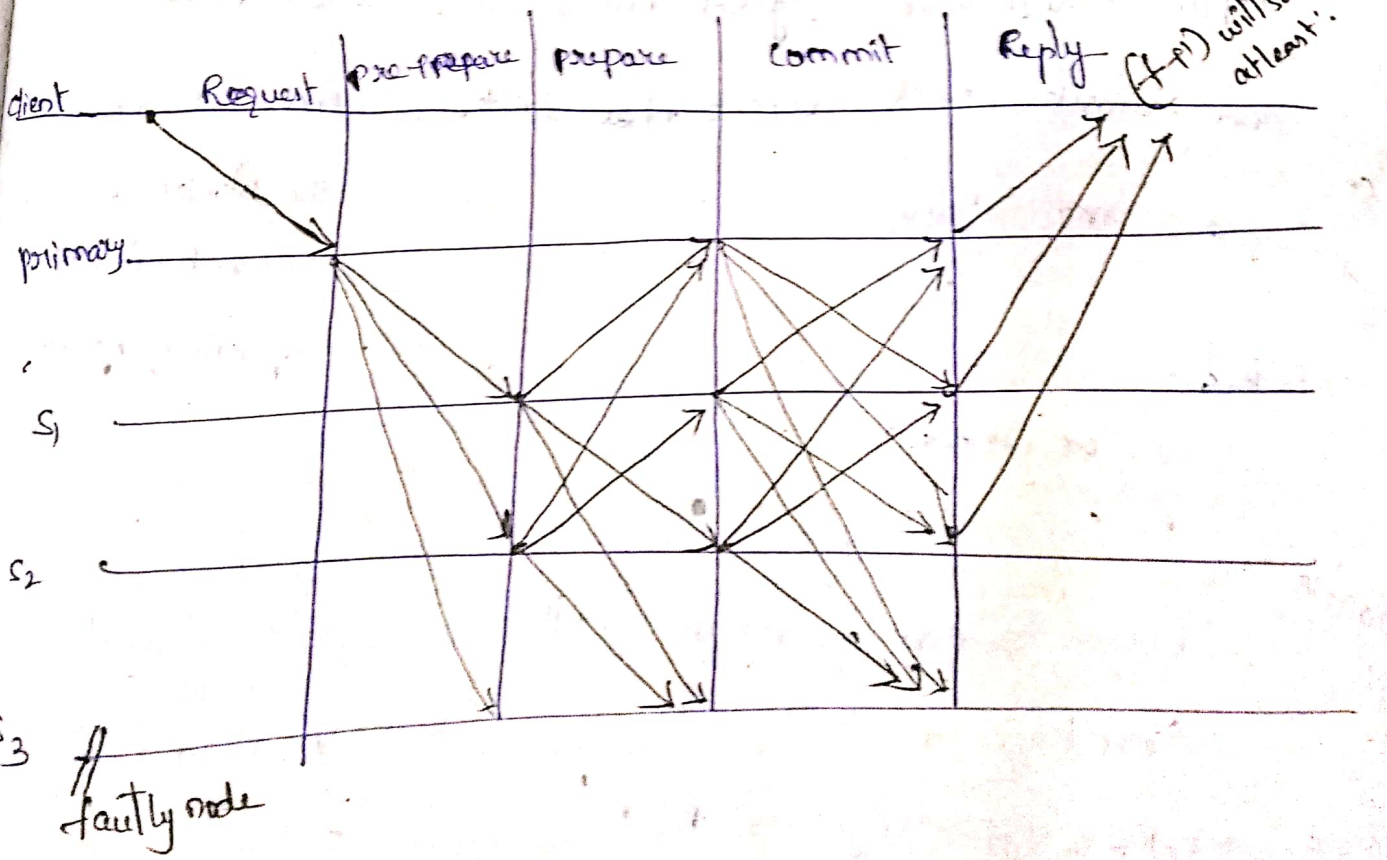
and prepare message are same, ie, same view, same seq.no, same digest, then we say no faulty node.

then that except the faulty node message, remaining nodes should be of (2f+1) majority good nodes then atleast f+1 nodes should have the majority to accept it.

If any node got crashed, it should be thought before the design of the the n/w. if we assume that 10 nodes will got crashed at max then the, n/w should atleast contain $3f+1+10$ where f-are faulty nodes.

Commit phase: $\langle commit, v, n, d, i \rangle_{\sigma_i}$





11/9/19 difference b/n permissioned & permissionless blockchains.

13/9/19 HSM - H/w security module
— used to store private keys & digital certificates.

→ ledger contains 2 datastructures
- Block chain
- world state

First block in chain is 'genesis' block

+ pair should have ledger + Transactions/ smart transactions. and . smart contract. when a new block is added then it is given to other peer database.

the blockchain as a new block(s) append modifications

~~Actors~~
~~BC Developer~~

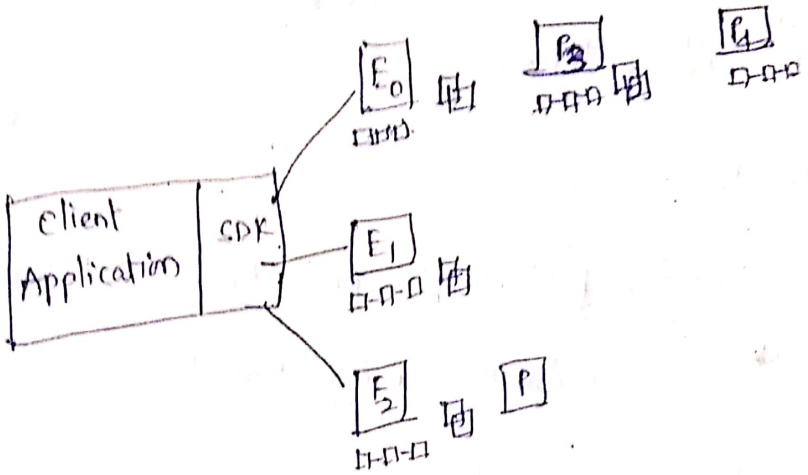
16/09/19 Linux started hyperledger project
- Hyper ledger fabric by IBM
- HFC - Hyperledger composer

→ Ordering - service
- it will follow some ordering algorithm and order transactions.

Endorsing peers → The peers who will accept or not.

committing peers → These peers will validate transaction.

Chaincode → smart contract.



Transaction flow

Endorse → order → validate.

Step①: propose the transaction.

client propose and send to endorsing peers.

step2: Execute proposed transaction.

step3: Proposal Response

- ie,, reply to client using their private key encrypted data.

Endorser policy: (P)

- E_0, E_1 and E_3 must sign
- P_3, P_4 are not part of the policy

Step 4: order transaction.

→ client orders ordering service

→ SOLO (single node, development)

→ Kafka (crash tolerant system)

- ordering service will prepare a block

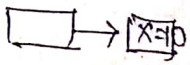
→ and it will sent to all peers.

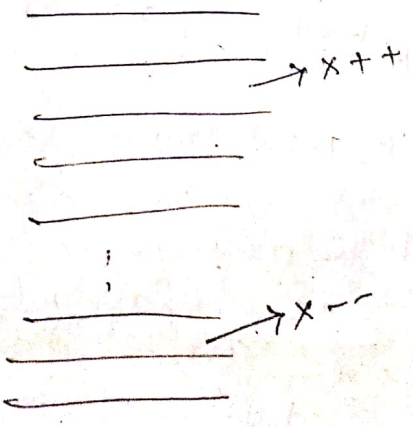
Step ⑤: Deliver transaction.

→ transfer to all the nodes (peers) including Endorsing & committing peers.

Validate based on

- Endorsing policy [ie, transaction is endorsed by all endorsers or not]
- double spending [Exactly two transactions are same].
- Variable modified by 2 or more transactions.





∵ x++ is not updated in the block chain, so, x-- will take 10 value.

x++ = 11

x-- = 9 inconsistency

Problem, so, only accept 1 transaction, and reject the other.

→ Note. Invalidate transactions are also placed in the block itself

Step ⑥ Vali

Step ⑦ N

Step ⑥ Validate transaction

→ add block to ~~∀~~ peer / node

Step ⑦ Notify Transaction

– notify to each & every client from node in the blockchain.