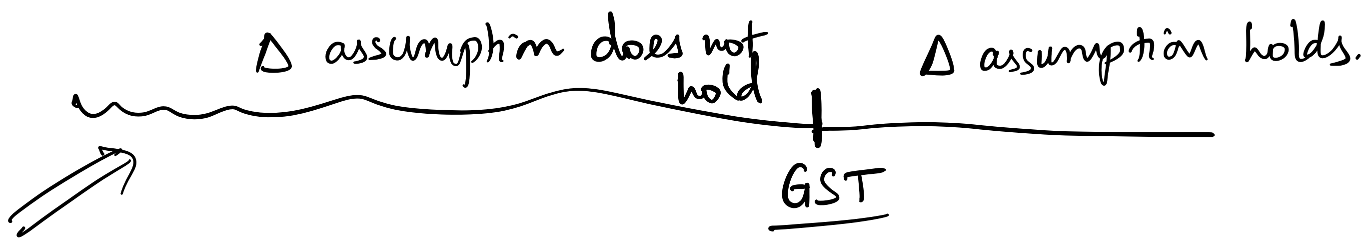
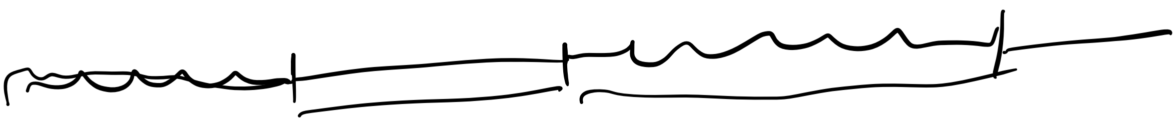
Partial sync: Periods of sync + async.

(DLS) ⟵ Dijkstra

1. Unknown Global stabilization time (GST): + known Δ:



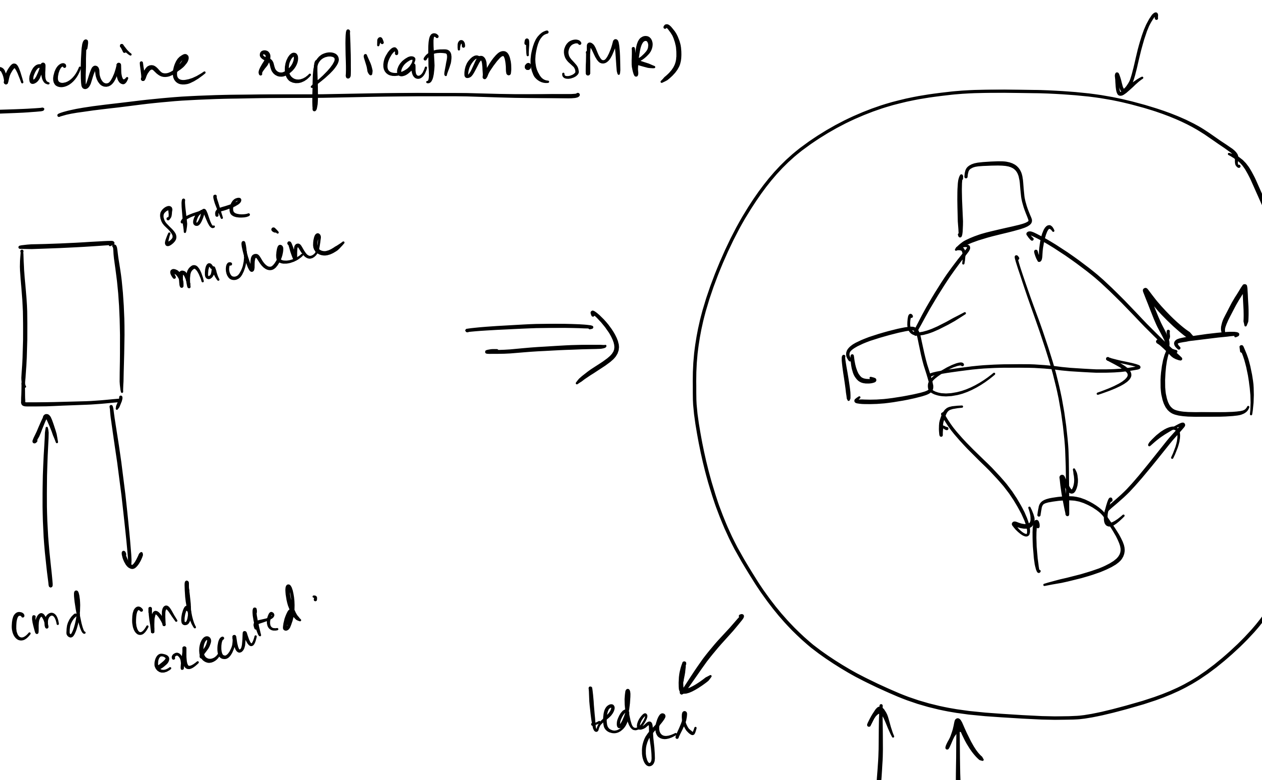
equivalent to



2. Unknown Δ:

Both defns are equivalent to each other.

State machine replication: (SMR)



Safety: Any two honest replicas to not output different cmds at the same pos. cmd_i cmd_j

Liveness: Every honest should eventually output a value sent by a client.

→ Agreement

→ termination & validity.

↳ One value vs a ledger of values. (ordering).

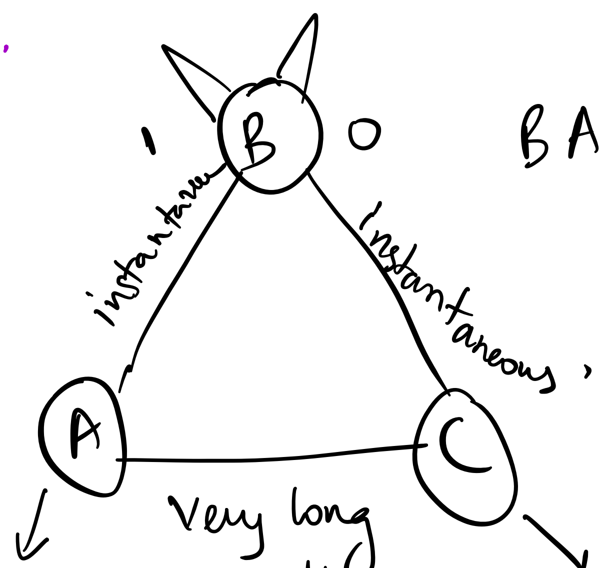
↳ Notion of clients: $t+1$ replicas.

BB: Dolev-Strong $t < n-1$ faults.

Partial synchrony (PS):

Under PS, it is impossible to tolerate $t \geq \frac{n}{3}$ Byzantine faults.

$t \geq \frac{n}{2}$ crash.

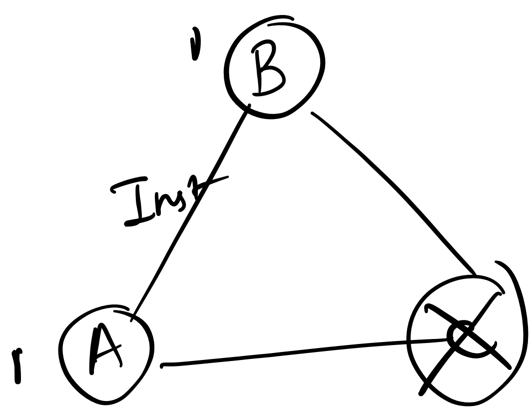


BA. "Split-brain attack"

time. 0

World 1: A & B are honest, C has crashed.
A & B start with input 1.

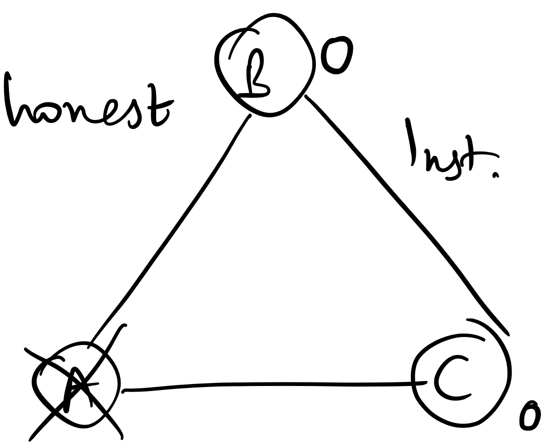
$t \geq \frac{n}{3}$ faults.



By validity constraint, A & B output 1.

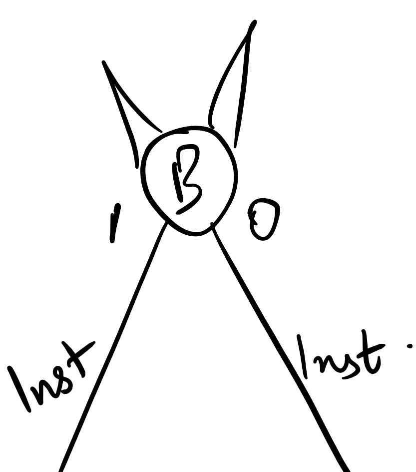
World 2:

B & C are honest



By validity constraint, B & C output 0.

World 3:



B is Byzantine
↳ brain 1 works as in

W1 with A

↳ 2, W2. 1

with C.

A: This world is exactly the same as world 1.

Observations:

- Setup assumptions.
- Existence of partial synchrony & the fault being Byzantine.

  partial synchrony ⟹ $n > 2t$.

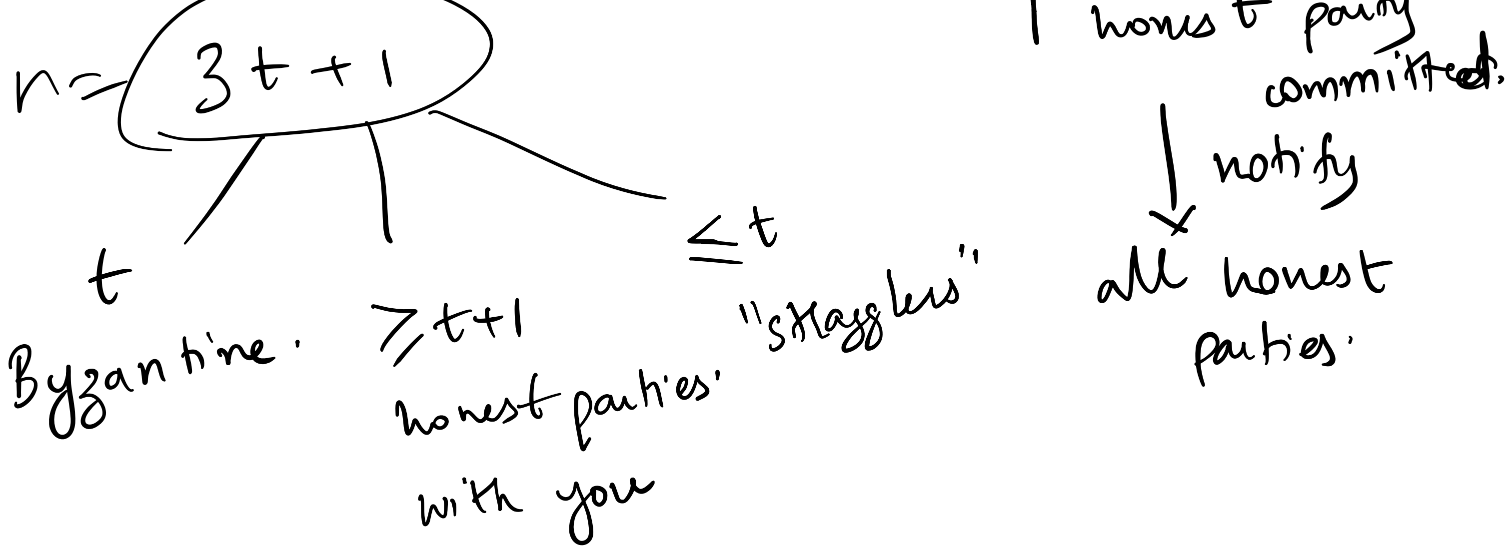
  Byzantine ⟹ clash. $n >$

  $t \geq \frac{n}{3}$ is not possible.

  $t < \frac{n}{3}$; $n = 3t + 1$

- If one honest party decides on a value, then all honest parties should decide on the same value.

  ↳ Not all honest parties can communicate with one another; we need a majority of the honest parties to "agree" on the same value.



"accept-commit" paradigm.

PBFT: Practical Byzantine Fault Tolerance

Castro & Liskov, 1999.
↓
Turing

Primary-backup paradigm:

"primary / leader: drive progress (liveness) in the system; order in which values are processed
replicas: ensuring safety holds.
& ensuring primary is doing its job correctly.

pros: In a good world: progress - streamlined.

cons: subtle attacks: ordering

Steady State (v): Replica i's perspective

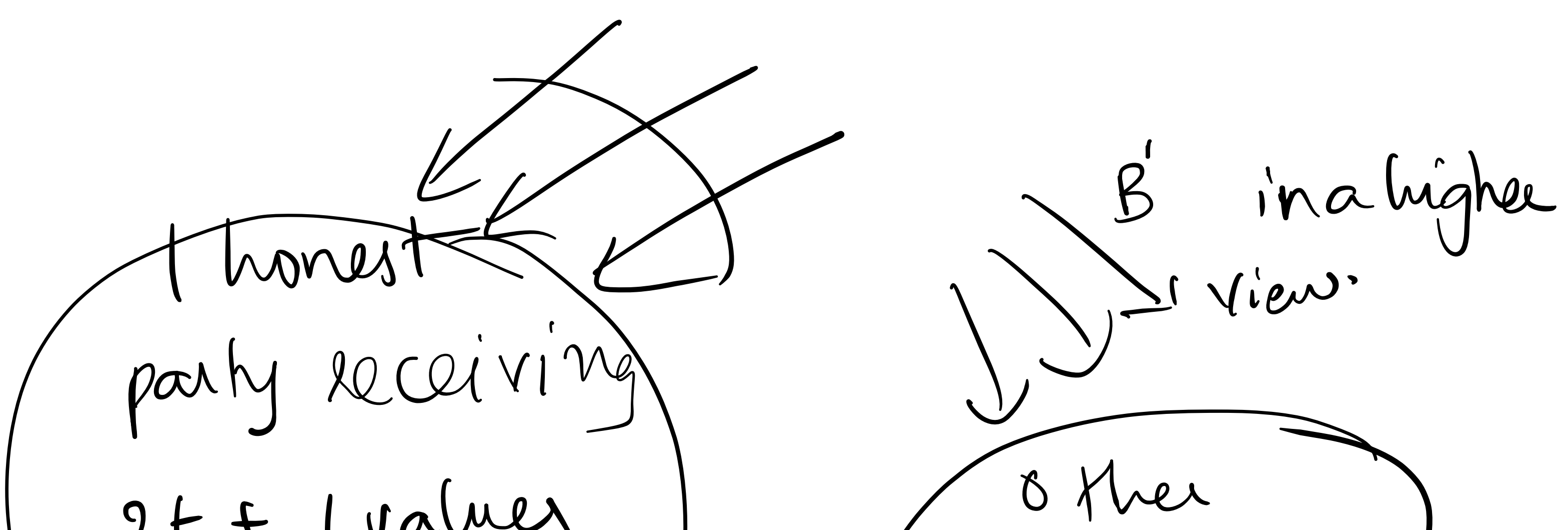
→ **Propose (pre-prepare):** Leader proposes <propose, B, v, ...>

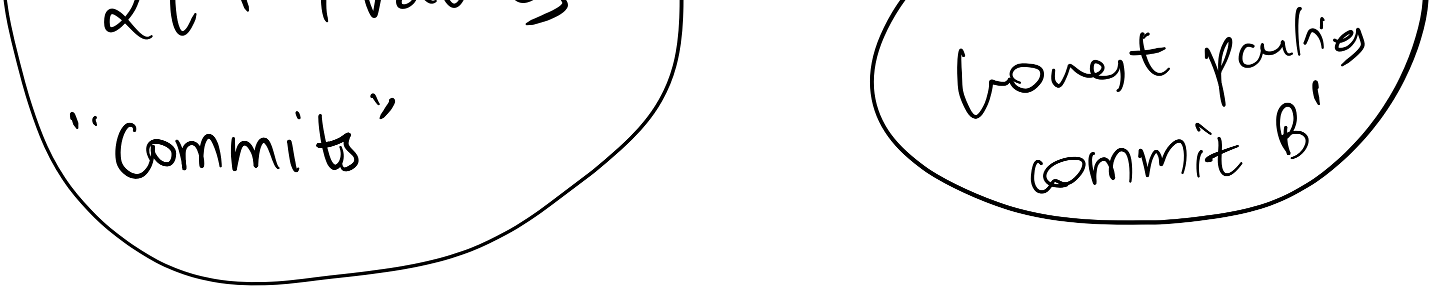
Vote1 (prepare): send vote <vote1, B, v, ...> "if it is safe" to vote for it.

Vote2 (commit): Replica i collects 2f+1 vote1's for the same value B "accepted / locked".

Replica i sends <vote2, B, v, 2f+1 vote1's>

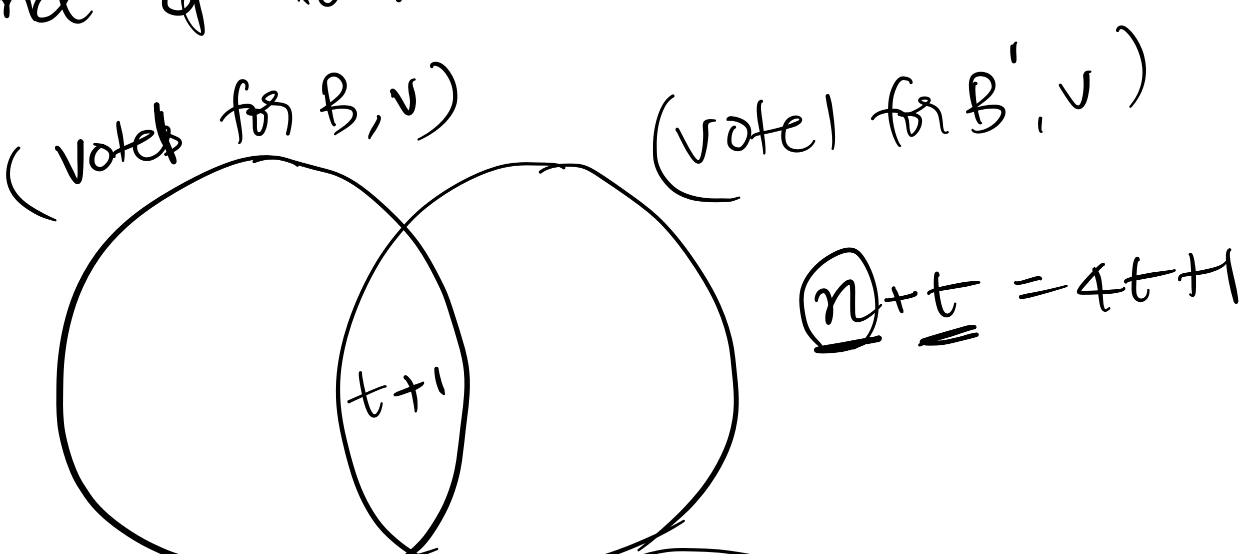
Commit (committed): Replica i collects 2f+1 vote2's for B then it commits B.





If an honest party commits B.
ie, it receives 2t+1 vote2 messages, then at least t+1 honest parties have receive a quorum of vote1 messages
(have locked on B)

Uniqueness within the view: first round of votes.



4t+2.

up to t parties will vote for both.

1. Only a unique value can be locked on in a given view.
2. If an honest party commits, the $\geq t+1$ honest parties lock on the committed value.

"Blame the leader" : View-change

Blame: send < "blame, v", <u>lock_i</u> > to the next leader

<u>New leader:</u>

On collecting 2t+1 of blame messages & start a new-view,

propose: pick a value with the

propose: ...
hi hest lock & propose it.

shows the 2t+1 locks it received.

<propose, B, v, status>

⇓

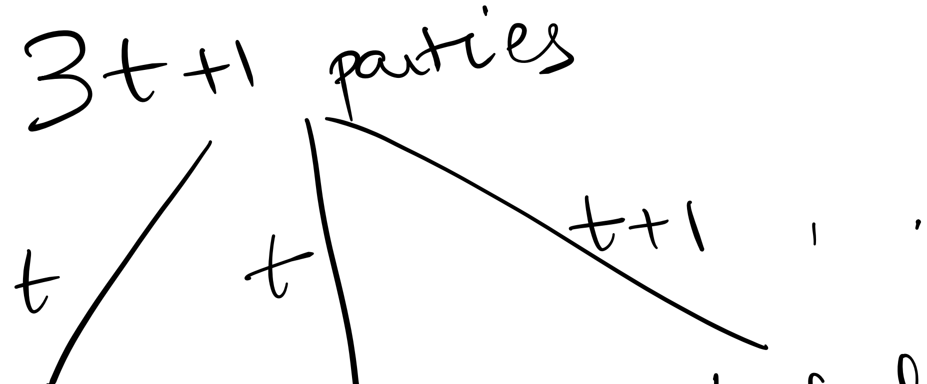
2t+1 locks. (1)

t+1 come from honest parties.

B, v

↓

t+1 honest parties hold locks. → ≥ 1 of these locks should be included in the status cert

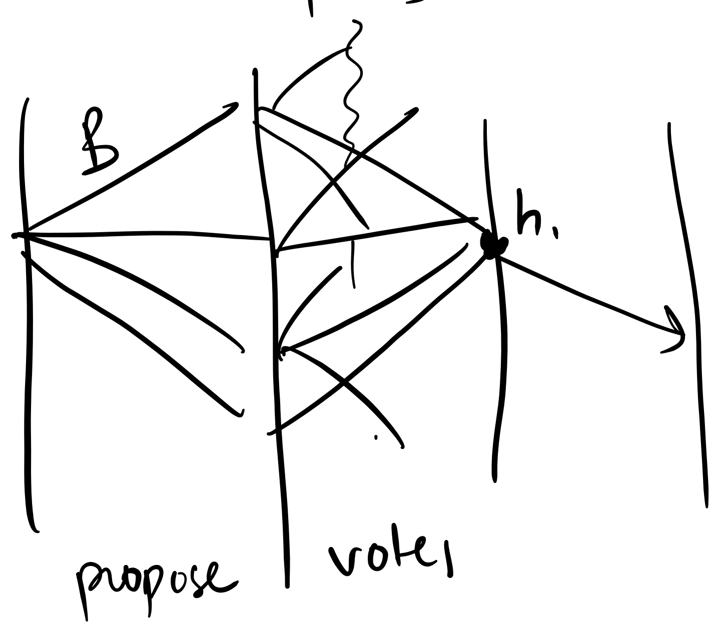


Byzantine stragglers. locked.

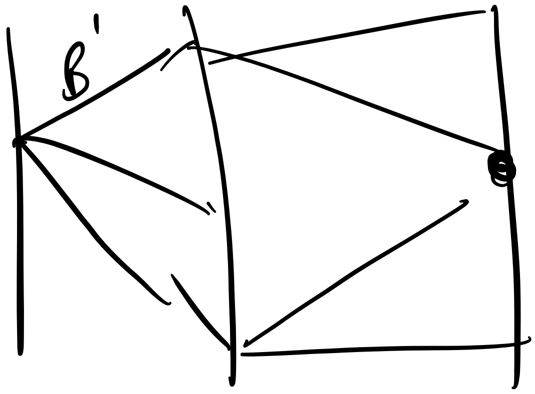
"if it is safe to do so":

Replica i votes if
- the value B is the same as the lock replica holds.
- if the proposed ~~is~~ value is B' ≠ B but there is a higher lock in the status.

View v : ~~Son~~ Honest party h_1 is locked(B, v)



View v+1 : h_1's status does not arrive



h_2 locked on $(B', v+1)$

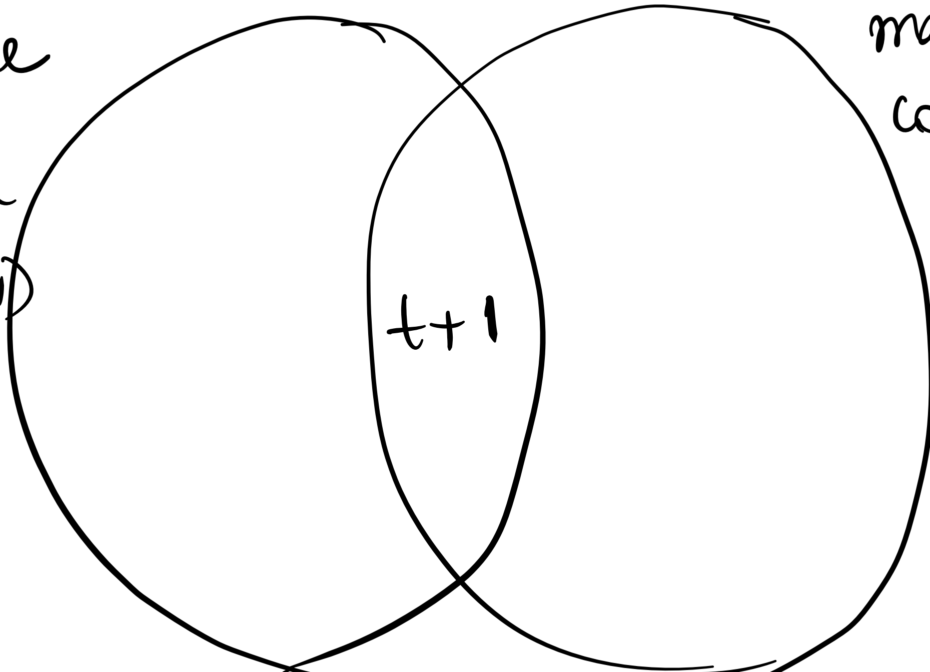
⋮

$v+t+1$

$(h_{t+1}$ on $B''\cdots, v+t+1)$

---

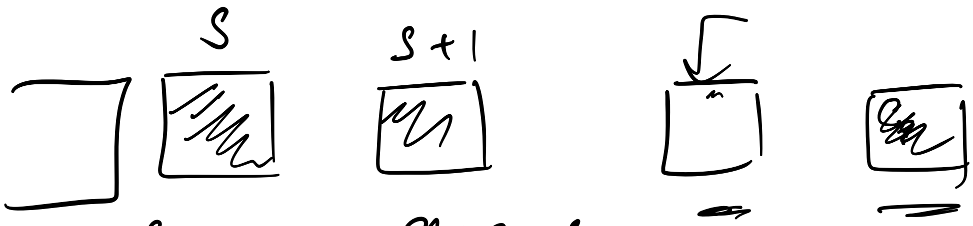
makes me commit a value (B, v)



makes my commit unsafe "status"

2t+1

· <u>One slot to many slots</u>:



propose propose

vote 1

vote2

commit.

↑

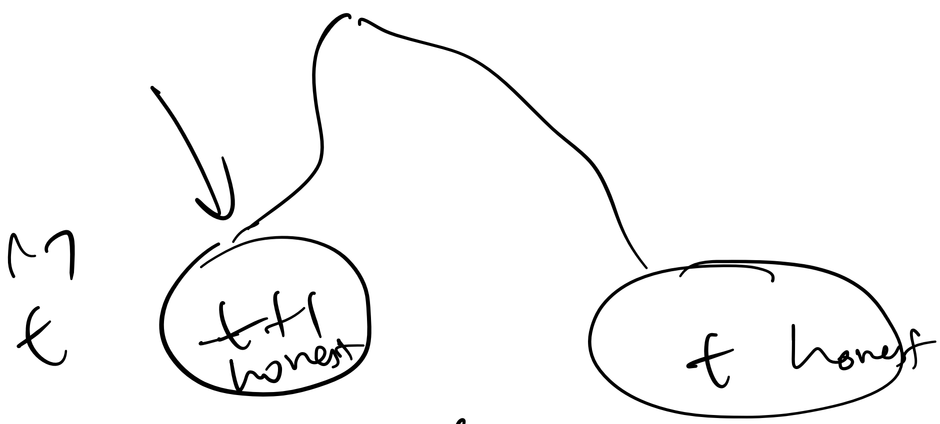
Byz parties

↳ Increase seqnos. to arbitrarily,

– lo_watermarks

– hi_watermark,

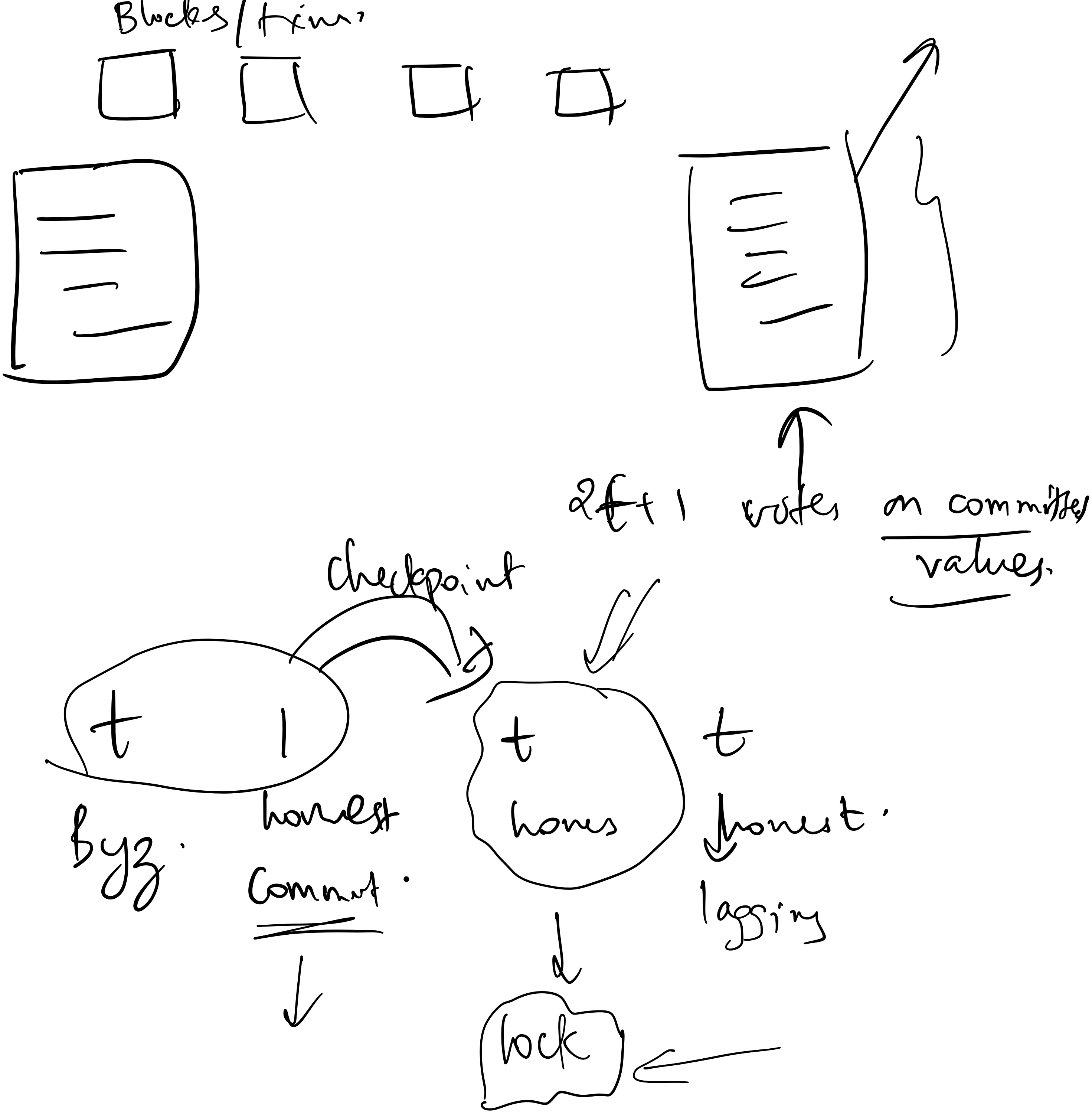
<u>Checkpointing</u>:



moved very fast

1million

lagging.



state transfer: (Dolev-Reiscuk) · n^2

Broadcast Extension protocols.

HotStuff / Casper / Tendermint : 2019

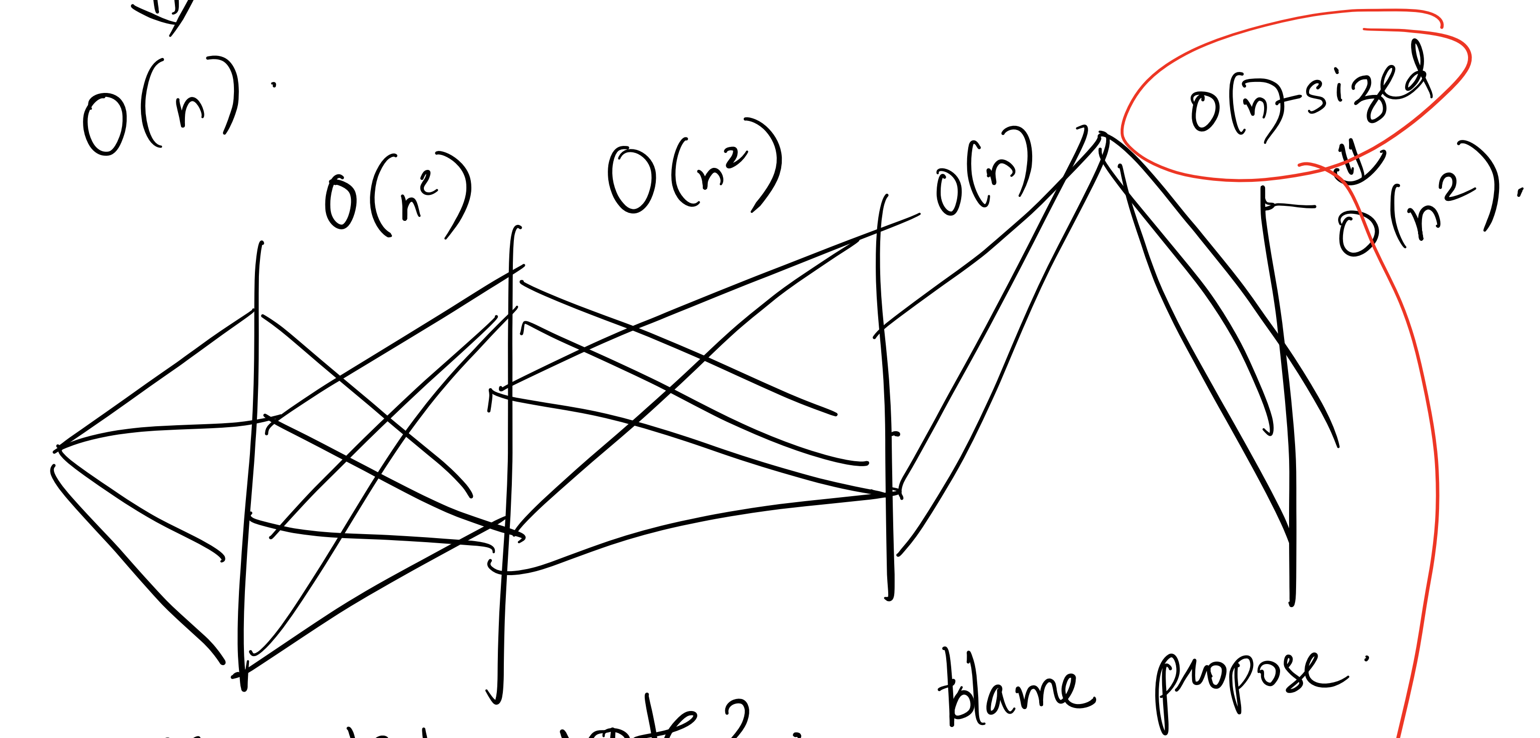
⇓ ⇓ ⇓

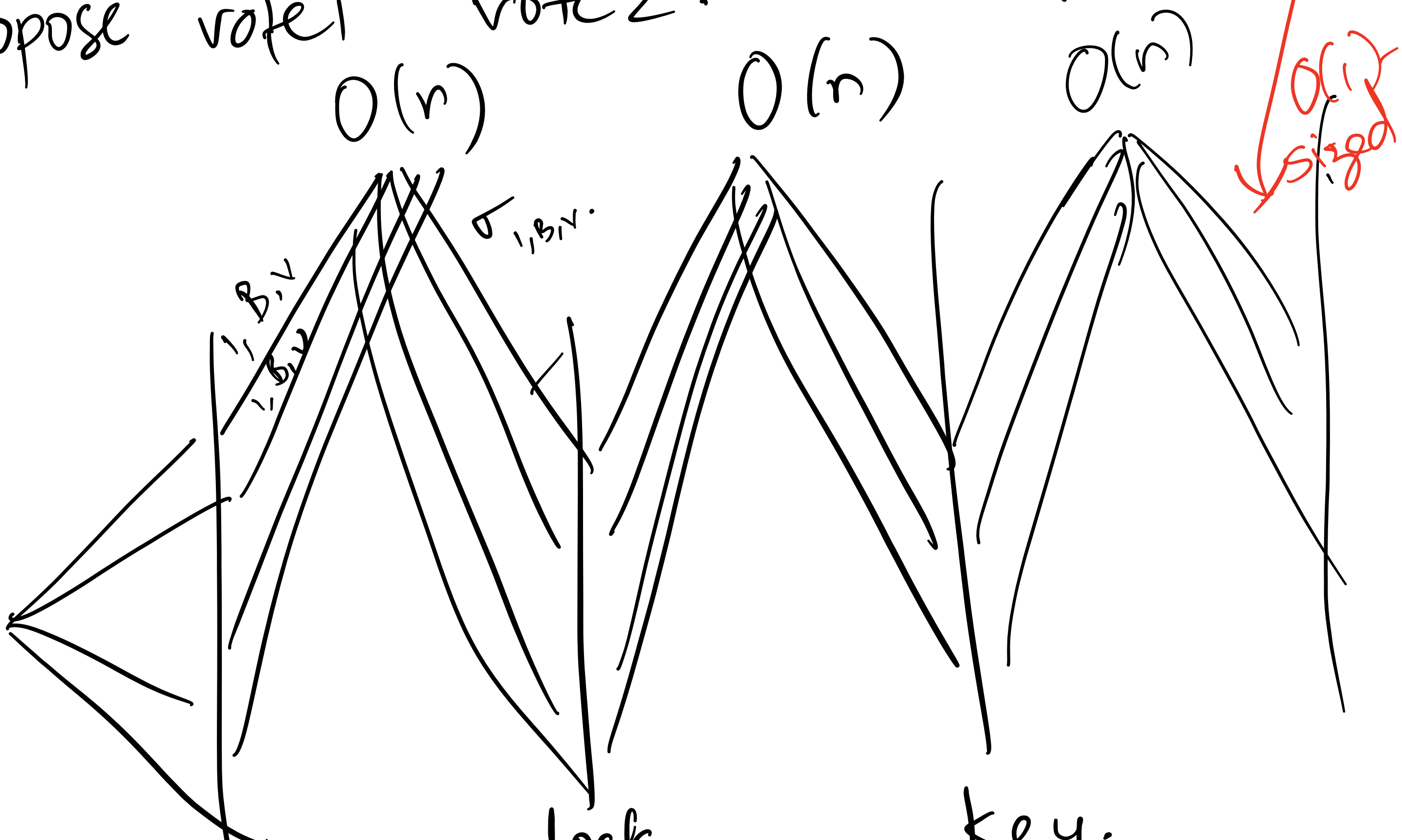
VMware Research, Ethereum

$O(n^2)$ communication within a view & during view-change.

⇓

$O(n)$.





Linear-view-change:

Safety: If t+1 honest parties had locked, status cert included at least one of them.

2t+1 ← (, O, , , ,) locks

⇓

"It is safe to vote".

Liveness: A party can unlock

Idea: Do not include a status cert, just include the highest lock.

Fix safety: No party will ever unlock.

quorum who guards my commit's safety.