ECS 265

The latest gossip on BFT consensus [Tendermint]

Ethan Buchman, Jae Kwon and Zarko Milosevic

Presented by : **SaiVivek Peddi & Suvi Varshney**

UCDAVIS
COMPUTER SCIENCE

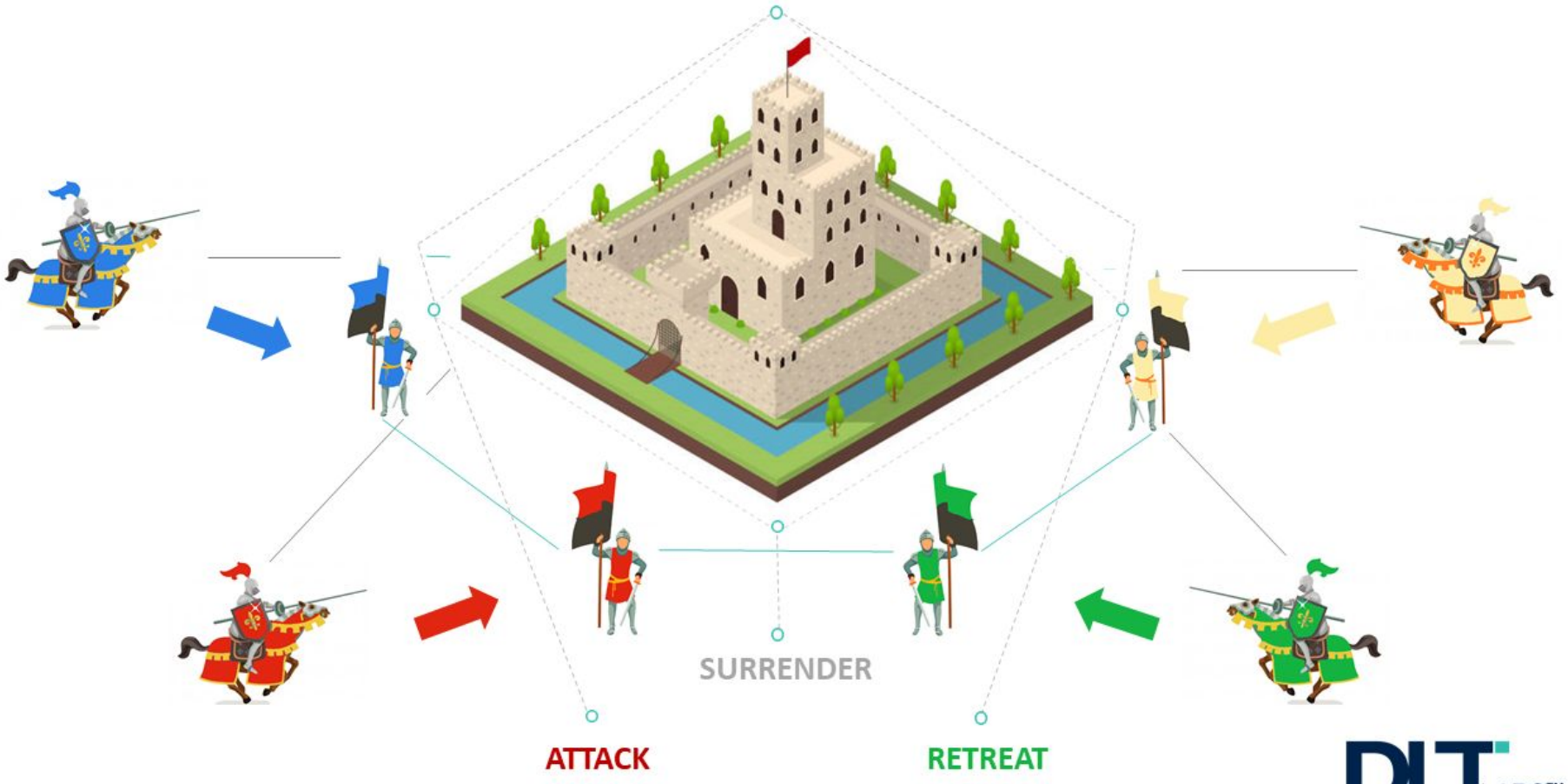
Table of Contents

UCDAVIS
COMPUTER SCIENCE

PBFT Questions

- What's Consensus in Synchronous and Asynchronous System?
- What's the difference between Synchronous and Asynchronous System?
- How and where the PBFT originate from?
- Why 3f + 1 replicas? Why not 2f + 1?
- What's the final outcome of PBFT (Abstraction of the Protocol)?
- Why 3 Phases in PBFT? Why not 2 Phases?
- What happens in each phase at a High Level?
- How does the election of primary happen?

Byzantine Generals' Problem



DLT LABS™

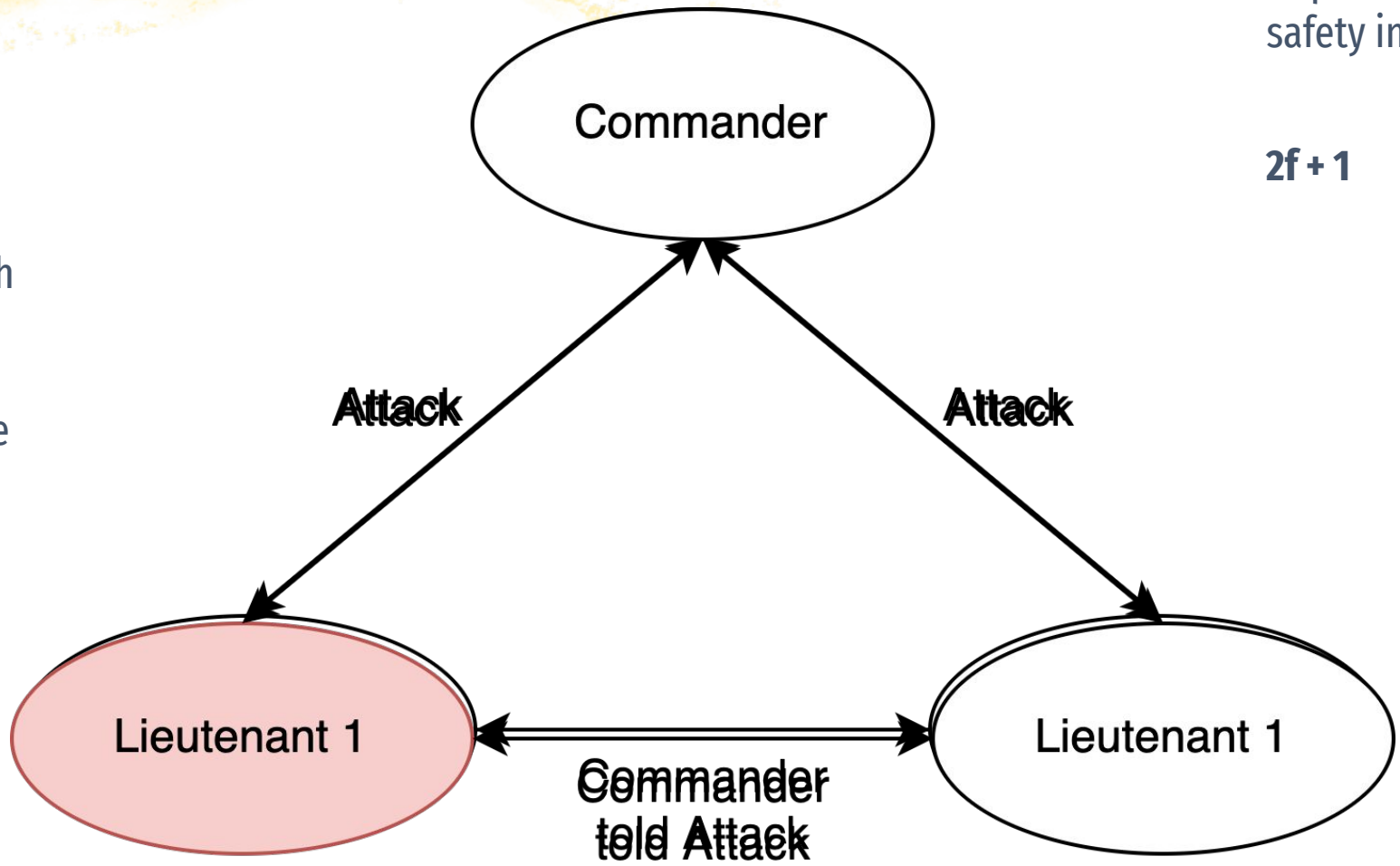
UCDAVIS COMPUTER SCIENCE

Scenarios - Sync

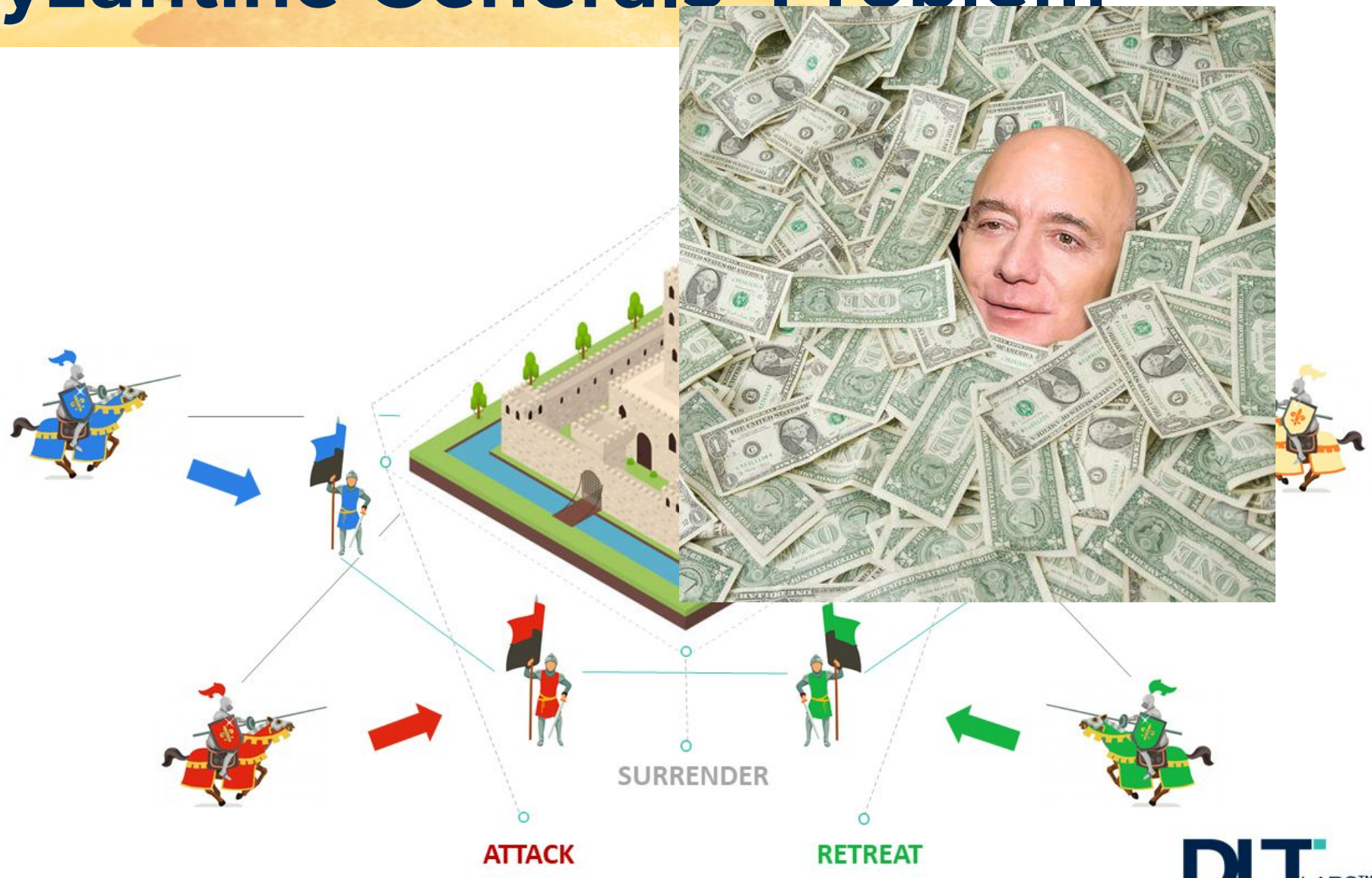
How many replicas are required to ensure safety in case of Sync?

2f + 1

In Sync, communication with non-faulty is guaranteed and everyone is genuine



Byzantine Generals' Problem



DLT LABS™

UCDAVIS
COMPUTER SCIENCE

Scenarios - Async

Consensus possible only if there are 4 if there is 1 Faulty Node.

For for 1 faulty node we need 3*1 + 1 nodes to reach consensus.

Generalizing this, for f faulty nodes we need 3*f + 1 nodes to reach consensus

Can we reach Consensus in either of the cases?

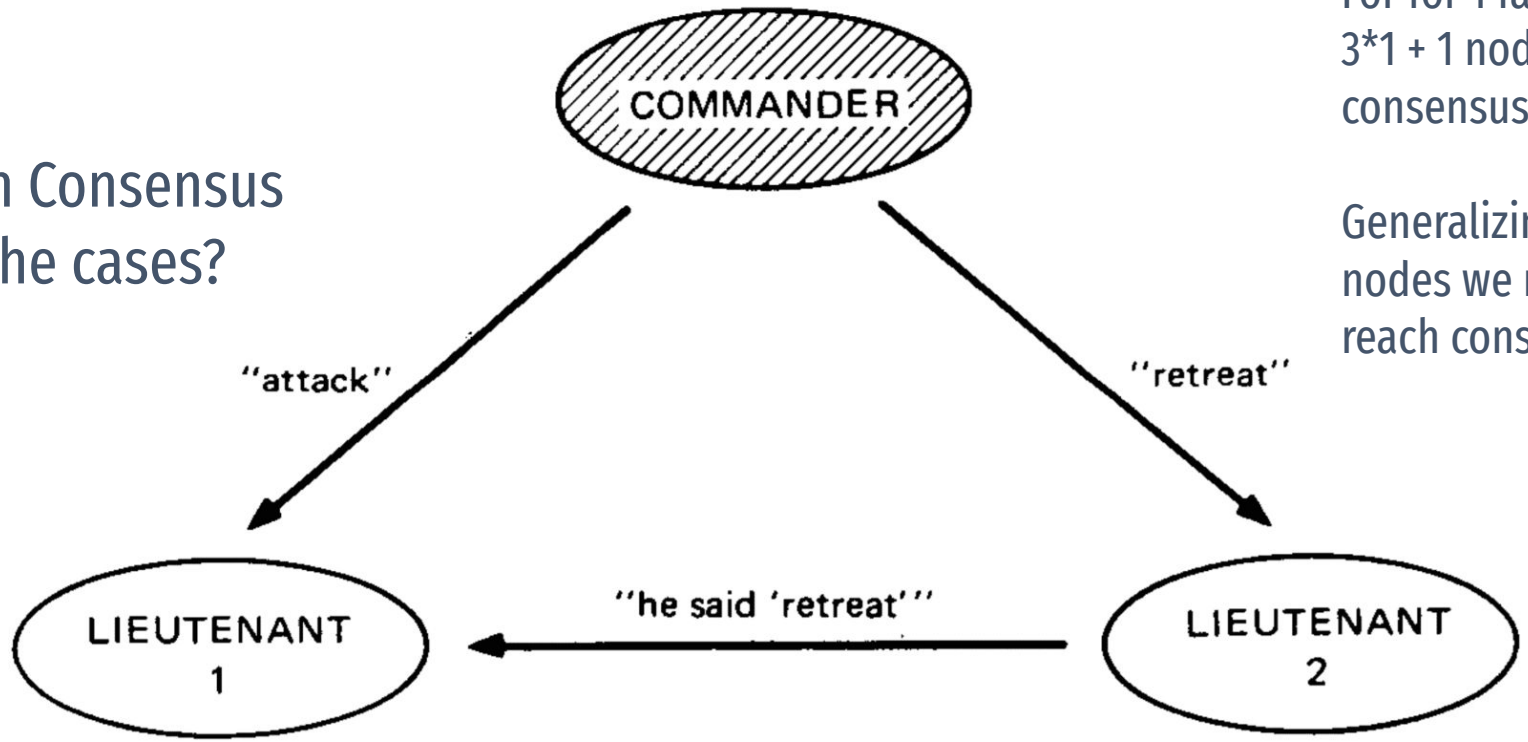
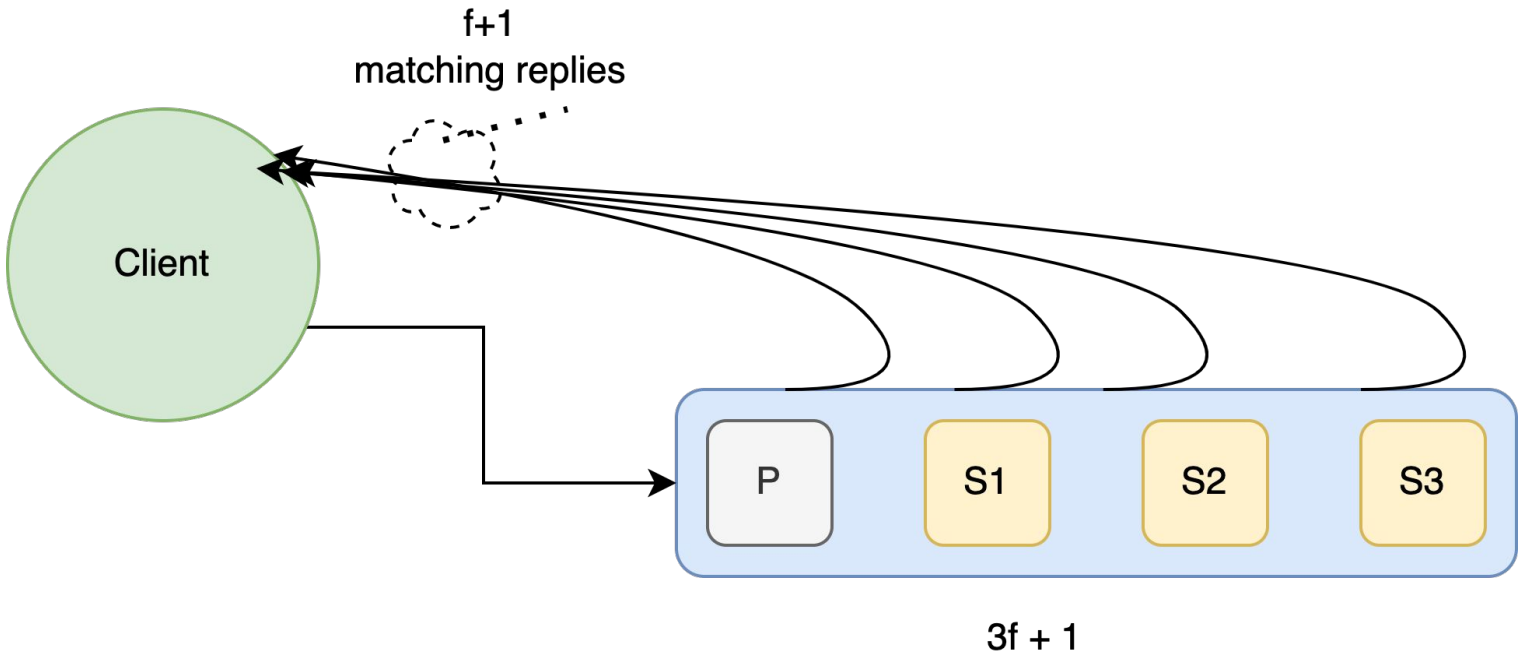


Fig. 2. The commander a traitor.

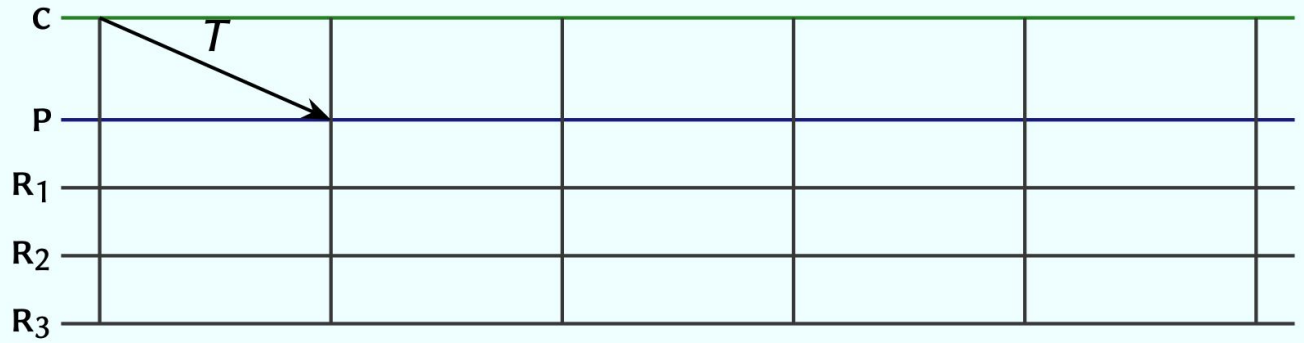
PBFT Questions

- What's Consensus in Synchronous and Asynchronous System?
- What's the difference between Synchronous and Asynchronous System?
- How and where the PBFT originate from?
- Why 3f + 1 replicas? Why not 2f + 1?
- What's the final outcome of PBFT (Abstraction of the Protocol)?
- Why 3 Phases in PBFT? Why not 2 Phases?
- What happens in each phase at a High Level?
- How does the election of primary happen?

PBFT in Abstract

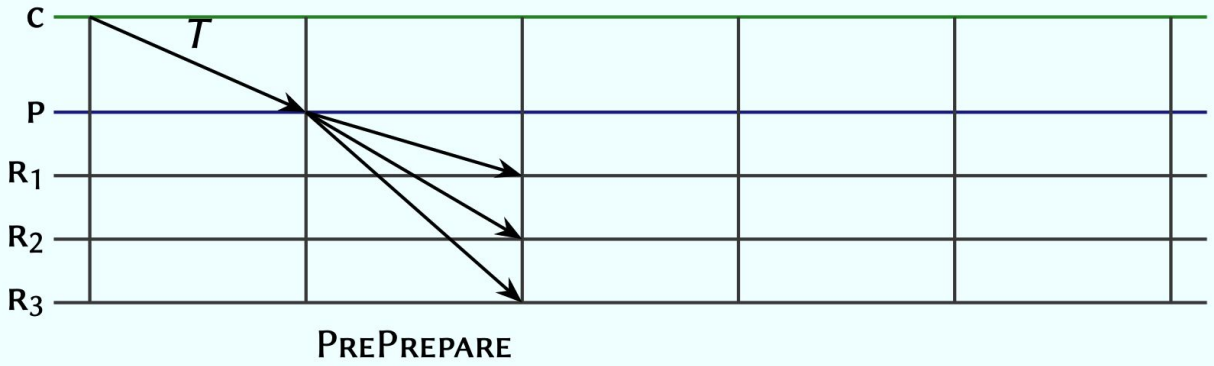


PBFT in Action



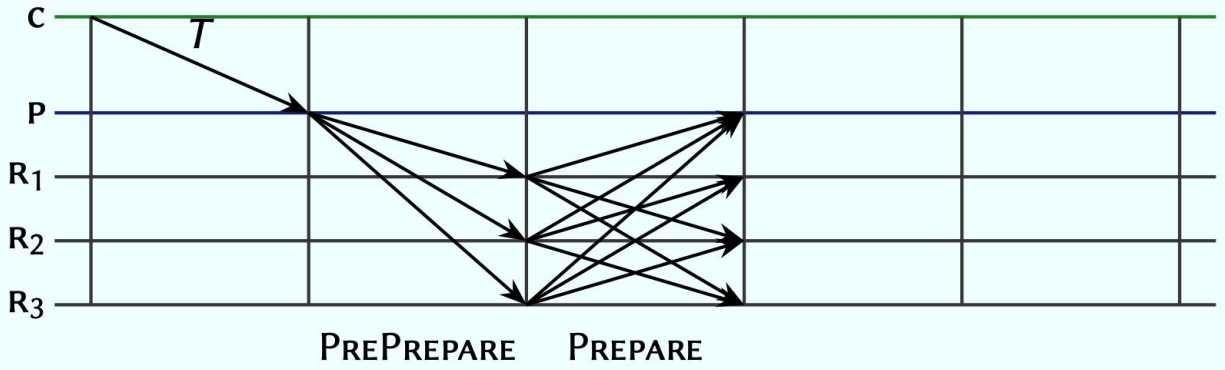
$\langle T \rangle_c$.

PBFT in Action



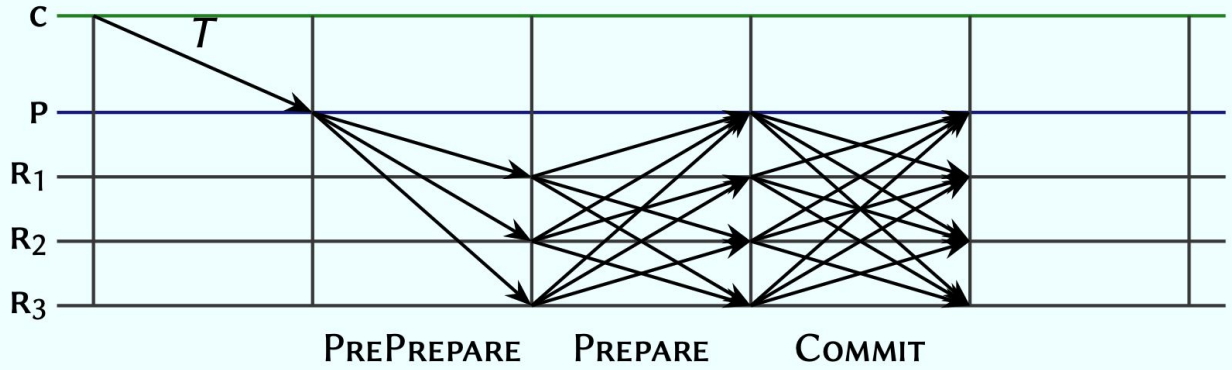
$\textsc{PrePrepare}(\langle T \rangle_c, v, \rho)$.

PBFT in Action



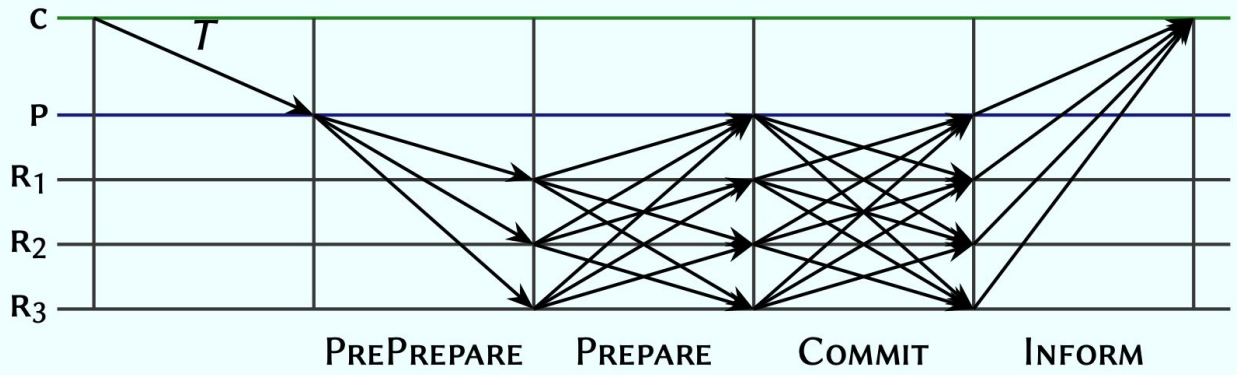
If receive PrePrepare message m: Prepare(m).

PBFT in Action



If **n** – **f** identical Prepare(m) messages: Commit(m).

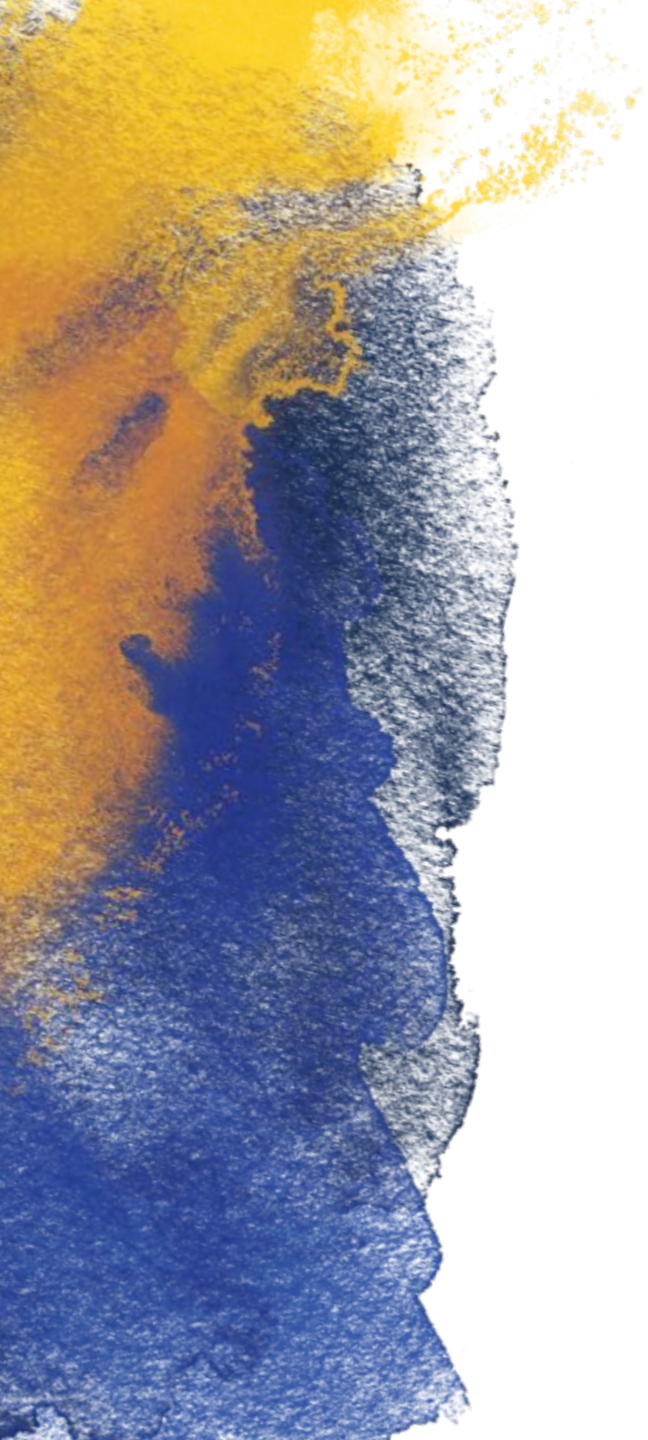
PBFT in Action



If **n** – **f** identical Commit(m) messages: execute, Inform($\langle T \rangle_c, \rho, r$).

PBFT Questions

- What's Consensus in Synchronous and Asynchronous System?
- What's the difference between Synchronous and Asynchronous System?
- How and where the PBFT originate from?
- Why 3f + 1 replicas? Why not 2f + 1?
- What's the final outcome of PBFT (Abstraction of the Protocol)?
- Why 3 Phases in PBFT? Why not 2 Phases?
- What happens in each phase at a High Level?
- How does the election of primary happen?



Tendermint

UCDAVIS
COMPUTER SCIENCE

Terms

- **State Machine Replication**
- Gossip protocol
- Block
- Height of a blockchain
- Round
- Byzantine Fault

Terms

- State Machine Replication
- **Gossip protocol**
- Block
- Height of a blockchain
- Round
- Byzantine Fault

Terms

- State Machine Replication
- Gossip protocol
- **Block**
- Height of a blockchain
- Round
- Byzantine Fault

Terms

- State Machine Replication
- Gossip protocol
- Block
- **Height of a blockchain**
- Round
- Byzantine Fault

Terms

- State Machine Replication
- Gossip protocol
- Block
- Height of a blockchain
- **Round**
- Byzantine Fault

Terms

- State Machine Replication
- Gossip protocol
- Block
- Height of a blockchain
- Round
- **Byzantine Fault**

UCDAVIS
COMPUTER SCIENCE

Overview of Tendermint

- What is Tendermint?
  - Byzantine Fault Tolerant Consensus
    - Like PBFT, 3 Phases
    - Rotation proposer
  - Tolerance of failure of upto ⅓ nodes/validators.

Overview of Tendermint

- What is Tendermint?
    - The 3 phrases of a single round are:

(Propose -> Pre-vote -> Pre-commit)

UCDAVIS
COMPUTER SCIENCE

Overview of Tendermint

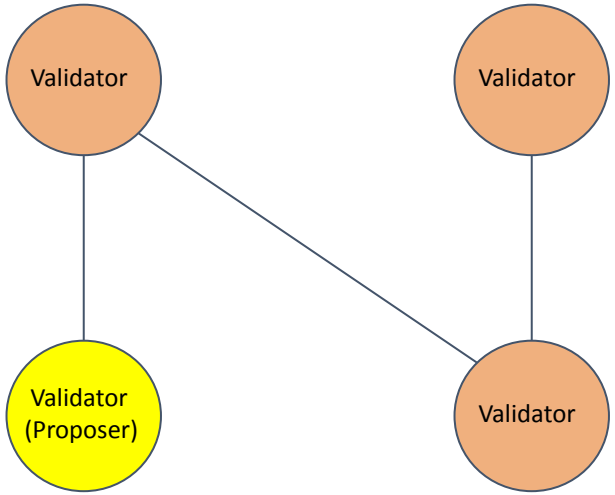
- What is Tendermint?
    - The 3 phrases of a single round are:

NewHeight -> (Propose -> Pre-vote -> Pre-commit)+ -> Commit -> NewHeight ->...

UCDAVIS
COMPUTER SCIENCE

Overview of Tendermint

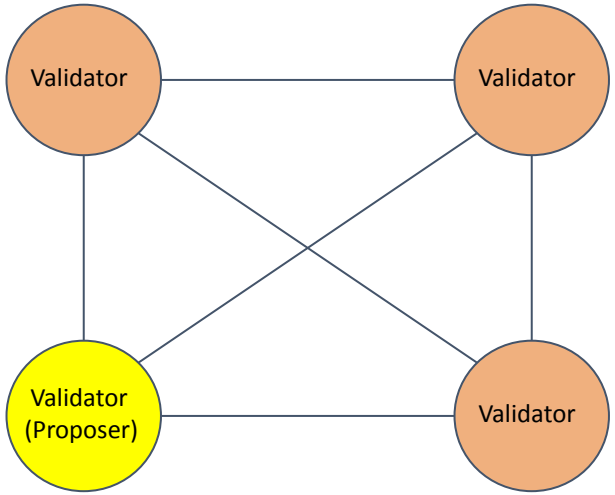
- What is Tendermint?



UCDAVIS
COMPUTER SCIENCE

Overview of Tendermint

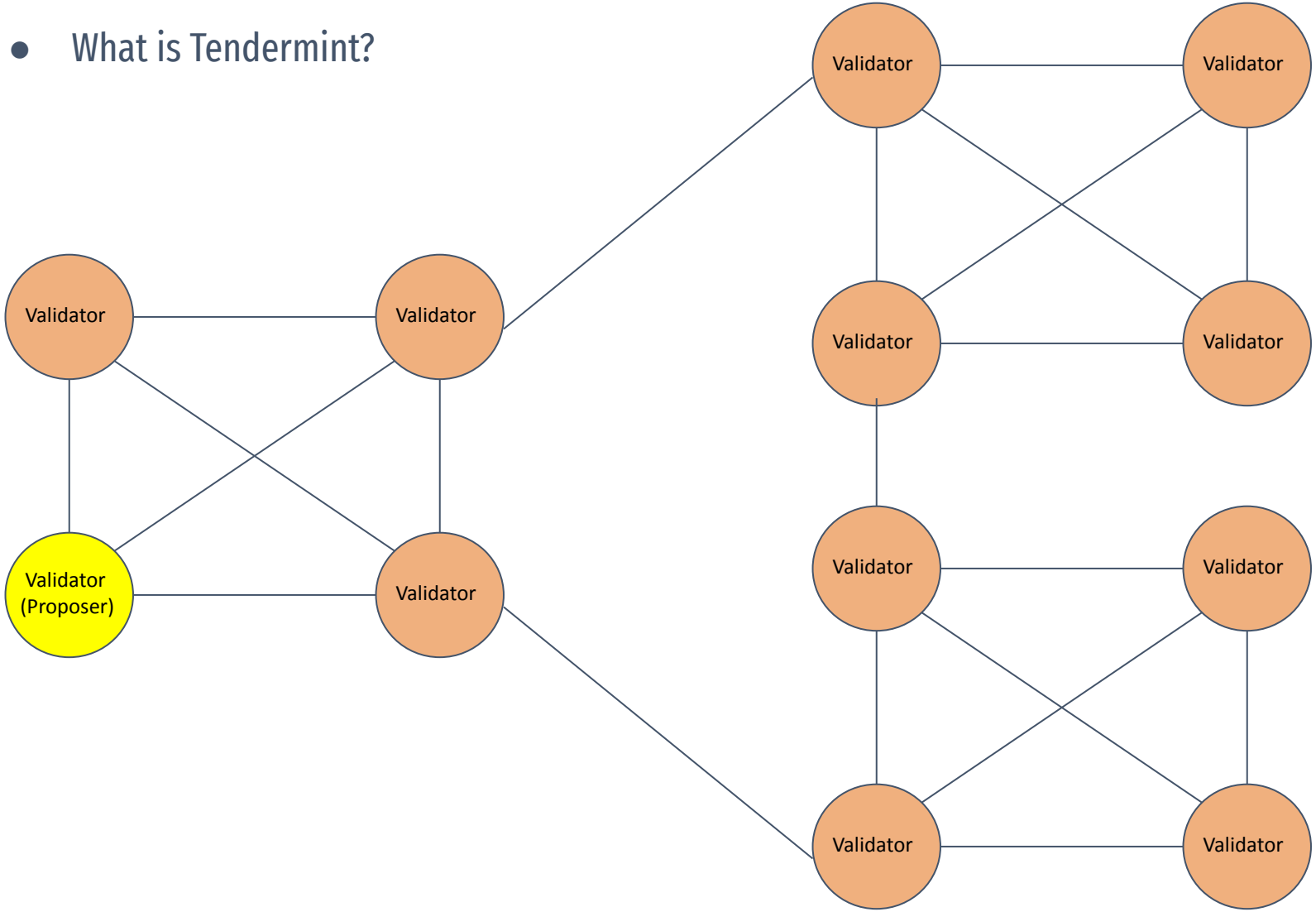
- What is Tendermint?



UCDAVIS
COMPUTER SCIENCE

Overview of Tendermint

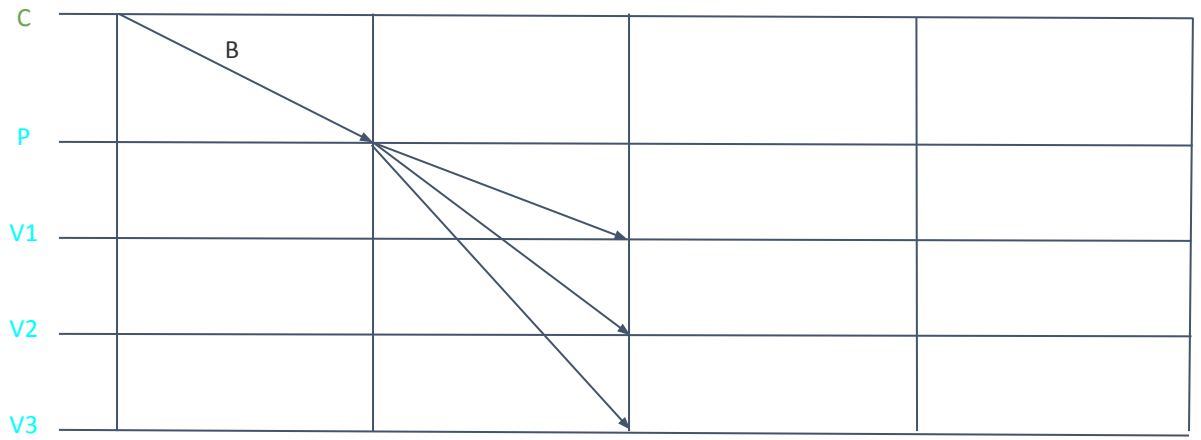
- What is Tendermint?



UCDAVIS
COMPUTER SCIENCE

Proposal Phase

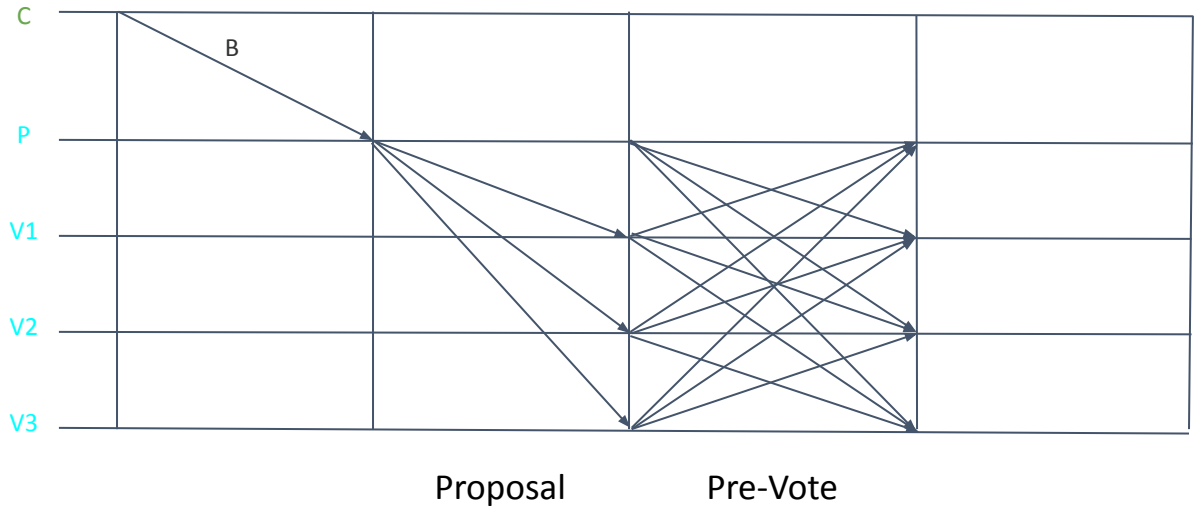
v, r, h



Proposal

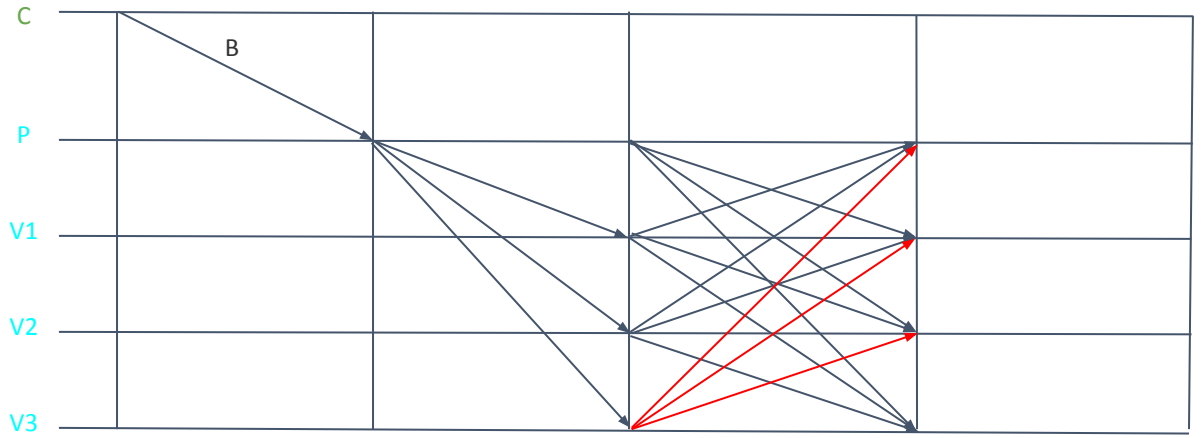
Pre-Vote Phase

v, r, h



Pre-Vote Phase

v, r, h



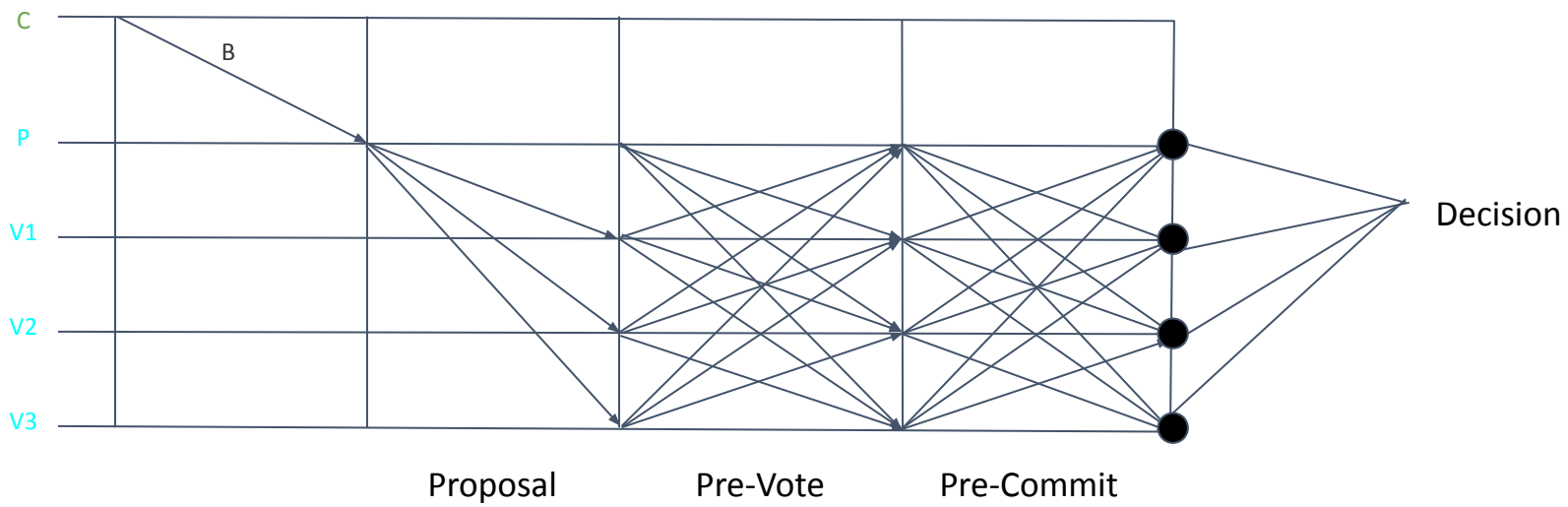
Proposal Pre-Vote

1. Failed validation
2. Proposal timeout

Proposal Timeout (r) = initTimeoutProposal + r ∗ timeoutDelta

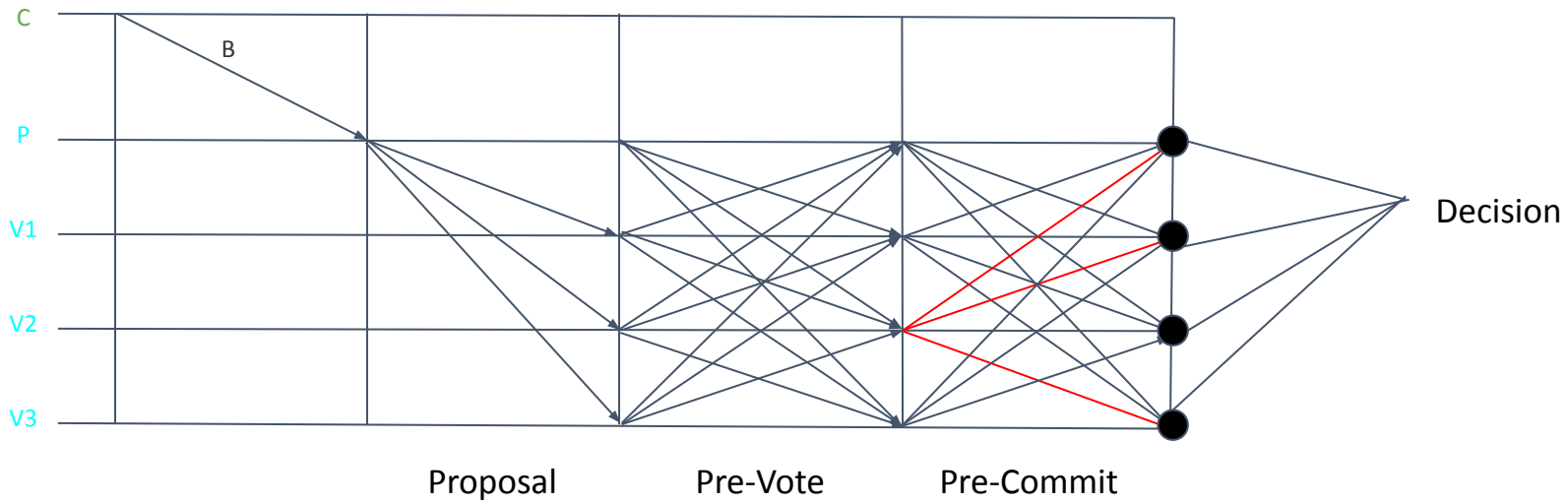
Pre-Commit Phase

v, r, h



Pre-Commit Phase

v, r, h



1. Failed validation of majority Pre-Vote
2. Pre-vote timeout before 2f+1 Pre-Vote

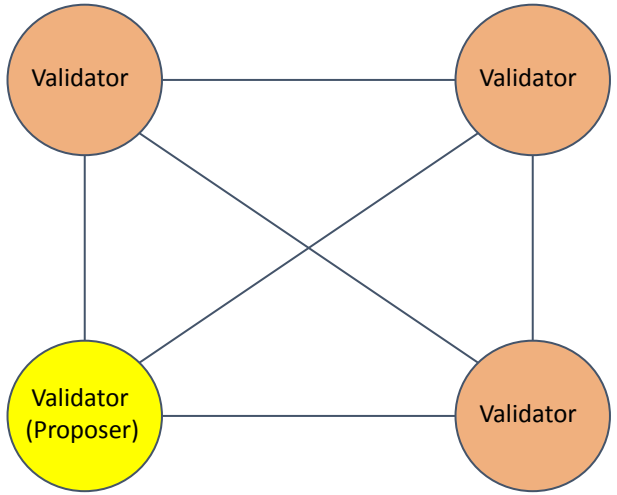
2f+1 Pre-Commit, f+1(majority in favour)

timeoutPre-Vote (r) = initTimeoutPre-Vote + r ∗ timeoutDelta

timeoutPre-Commit (r) = initTimeoutPre-Commit + r ∗ timeoutDelta

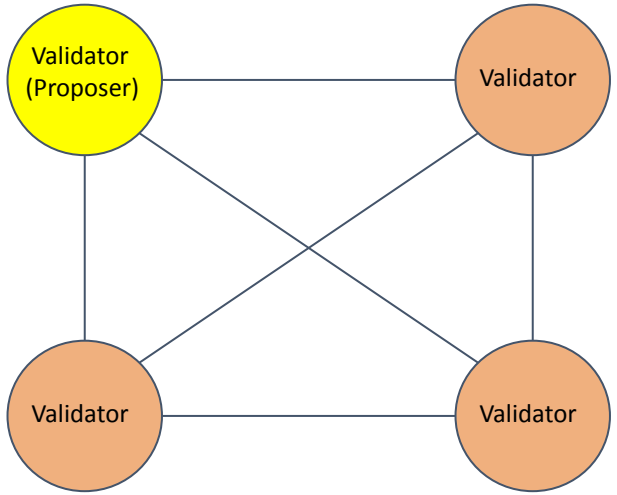
UCDAVIS COMPUTER SCIENCE

View Change/Faulty Proposer



View Change

- The round is complete and all the non-faulty nodes committed.
- The proposer is faulty. (offline, timeout, bad proposal)
- proposer(h, round) ←------ External Function
- Total Votes = n = 3f + 1 (Can be divided in a discriminatory way)



UCDAVIS
COMPUTER SCIENCE

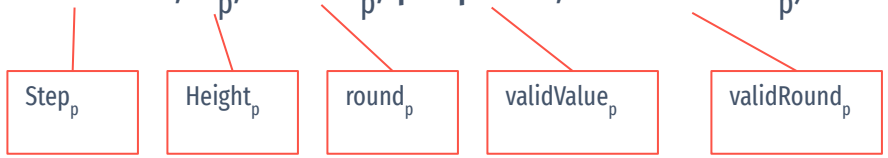
Algorithm

- $Step_p$: {**propose**, **prevote**, **precommit**}
- $lockedValue_p$: {**nil**, *value of transaction*}
- $lockedRound_p$: {**-1**, *round number*}
- $validValue_p$: {**nil**, *value of transaction*}
- $validRound_p$: {**-1**, *round number*}
- h_p : {**0**, *height of the blockchain*}
- $decision_p$: {**nil**, *log of activity*}

UCDAVIS
COMPUTER SCIENCE

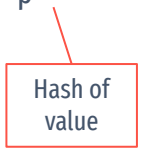
Difference between messages

- Proposal: ⟨PROPOSAL, h_p, $round_p$, proposal, $validRound_p$⟩



- Pre-Vote: ⟨PREVOTE, h_p, $round_p$, id(v)⟩

- Pre-Commit: ⟨PRECOMMIT, h_p, $round_p$, id(v))⟩



id(v) = id(v')

v = v'

UCDAVIS COMPUTER SCIENCE

Algorithm

itialization:
$h_p := 0$
$round_p := 0$
$step_p \in \{propose, prevote, precommit\}$
$decision_p[] := nil$
$lockedValue_p := nil$
$lockedRound_p := -1$
$validValue_p := nil$
$validRound_p := -1$

upon start **do** $StartRound(0)$

Function $StartRound(round)$ **:**
  $round_p \leftarrow round$
  $step_p \leftarrow propose$
  if proposer$(h_p, round_p) = p$ **then**
    if $validValue_p \neq nil$ **then**
      $proposal \leftarrow validValue_p$
    else
      $proposal \leftarrow getValue()$
    broadcast $\langle$PROPOSAL$, h_p, round_p, proposal, validRound_p\rangle$
  else
    schedule $OnTimeoutPropose(h_p, round_p)$ to be executed **after** $timeoutPropose(round_p)$

Function $OnTimeoutPropose(height, round)$ **:**
  if $height = h_p \wedge round = round_p \wedge step_p = propose$ **then**
    broadcast $\langle$PREVOTE$, h_p, round_p, nil\rangle$
    $step_p \leftarrow prevote$

UCDAVIS
COMPUTER SCIENCE

Algorithm

upon $\langle \text{PROPOSAL}, h_p, round_p, v, -1 \rangle$ **from** proposer$(h_p, round_p)$ **while** $step_p = propose$ **do**
  if $valid(v) \wedge (lockedRound_p = -1 \vee lockedValue_p = v)$ **then**
    broadcast $\langle \text{PREVOTE}, h_p, round_p, id(v) \rangle$
  else
    broadcast $\langle \text{PREVOTE}, h_p, round_p, nil \rangle$
  $step_p \leftarrow prevote$

upon $\langle \text{PROPOSAL}, h_p, round_p, v, vr \rangle$ **from** proposer$(h_p, round_p)$ **AND** $2f + 1$ $\langle \text{PREVOTE}, h_p, vr, id(v) \rangle$ **while** $step_p = propose \wedge (vr \geq 0 \wedge vr < round_p)$ **do**
  if $valid(v) \wedge (lockedRound_p \leq vr \vee lockedValue_p = v)$ **then**
    broadcast $\langle \text{PREVOTE}, h_p, round_p, id(v) \rangle$
  else
    broadcast $\langle \text{PREVOTE}, h_p, round_p, nil \rangle$
  $step_p \leftarrow prevote$

upon $2f + 1$ $\langle \text{PREVOTE}, h_p, round_p, * \rangle$ **while** $step_p = prevote$ for the first time **do**
  schedule $OnTimeoutPrevote(h_p, round_p)$ to be executed **after** $timeoutPrevote(round_p)$

Function $OnTimeoutPrevote(height, round)$ **:**
  if $height = h_p \wedge round = round_p \wedge step_p = prevote$ **then**
    broadcast $\langle \text{PRECOMMIT}, h_p, round_p, nil \rangle$
    $step_p \leftarrow precommit$

UCDAVIS COMPUTER SCIENCE

Algorithm

upon $\langle \text{PROPOSAL}, h_p, round_p, v, * \rangle$ **from** proposer$(h_p, round_p)$ **AND** $2f + 1$ $\langle \text{PREVOTE}, h_p, round_p, id(v) \rangle$ **while** $valid(v) \wedge step_p \geq prevote$ for the first time **do**
  if $step_p = prevote$ **then**
    $lockedValue_p \leftarrow v$
    $lockedRound_p \leftarrow round_p$
    broadcast $\langle \text{PRECOMMIT}, h_p, round_p, id(v) \rangle$
    $step_p \leftarrow precommit$
  $validValue_p \leftarrow v$
  $validRound_p \leftarrow round_p$

upon $2f + 1$ $\langle \text{PREVOTE}, h_p, round_p, nil \rangle$ **while** $step_p = prevote$ **do**
  broadcast $\langle \text{PRECOMMIT}, h_p, round_p, nil \rangle$
  $step_p \leftarrow precommit$

upon $2f + 1$ $\langle \text{PRECOMMIT}, h_p, round_p, * \rangle$ for the first time **do**
  schedule $OnTimeoutPrecommit(h_p, round_p)$ to be executed **after** $timeoutPrecommit(round_p)$

Function $OnTimeoutPrecommit(height, round)$ **:**
  if $height = h_p \wedge round = round_p$ **then**
    $StartRound(round_p + 1)$

UCDAVIS
COMPUTER SCIENCE

Algorithm

upon $\langle \text{PROPOSAL}, h_p, r, v, * \rangle$ **from** proposer(h_p, r) **AND** $2f + 1$ $\langle \text{PRECOMMIT}, h_p, r, id(v) \rangle$ **while** $decision_p[h_p] = nil$ **do**
  if $valid(v)$ **then**
    $decision_p[h_p] = v$
    $h_p \leftarrow h_p + 1$
    reset $lockedRound_p$, $lockedValue_p$, $validRound_p$ and $validValue_p$ to initial values and empty message log
    $StartRound(0)$

upon $f + 1$ $\langle *, h_p, round, *, * \rangle$ **with** $round > round_p$ **do**
  $StartRound(round)$

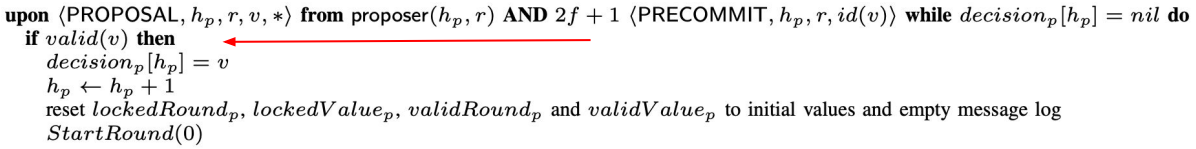
UCDAVIS
COMPUTER SCIENCE

Byzantine Fault tolerance

- n = 3f+1 nodes/validators
- 2f+1 nodes online/non faulty (1/3 rd)
- f+1 needed for majority
- 4 nodes

Proof of Lemmas

- **Lemma 1.** For all f ≥ 0, any two sets of processes with voting power at least equal to 2f + 1 have at least one correct process in common.
- **Lemma 2.** If f + 1 correct processes lock value v in round r0 (lockedValue = v and lockedRound = r0), then in all rounds r > r0, they send PREVOTE for id(v) or nil.
- **Lemma 3.** Tendermint Algorithm satisfies Agreement.
- **Lemma 4.** Tendermint Algorithm satisfies Validity.
- **Lemma 5.** Tendermint Algorithm satisfies Termination.



UCDAVIS COMPUTER SCIENCE

Time of execution & Complexity

- n nodes send messages to all the n-1 nodes, n^2 complexity
- Lab conditions: 1000 transactions per second, (network of 16 fully connected nodes)
- Real conditions: 10 transactions per second
- Increases exponentially.

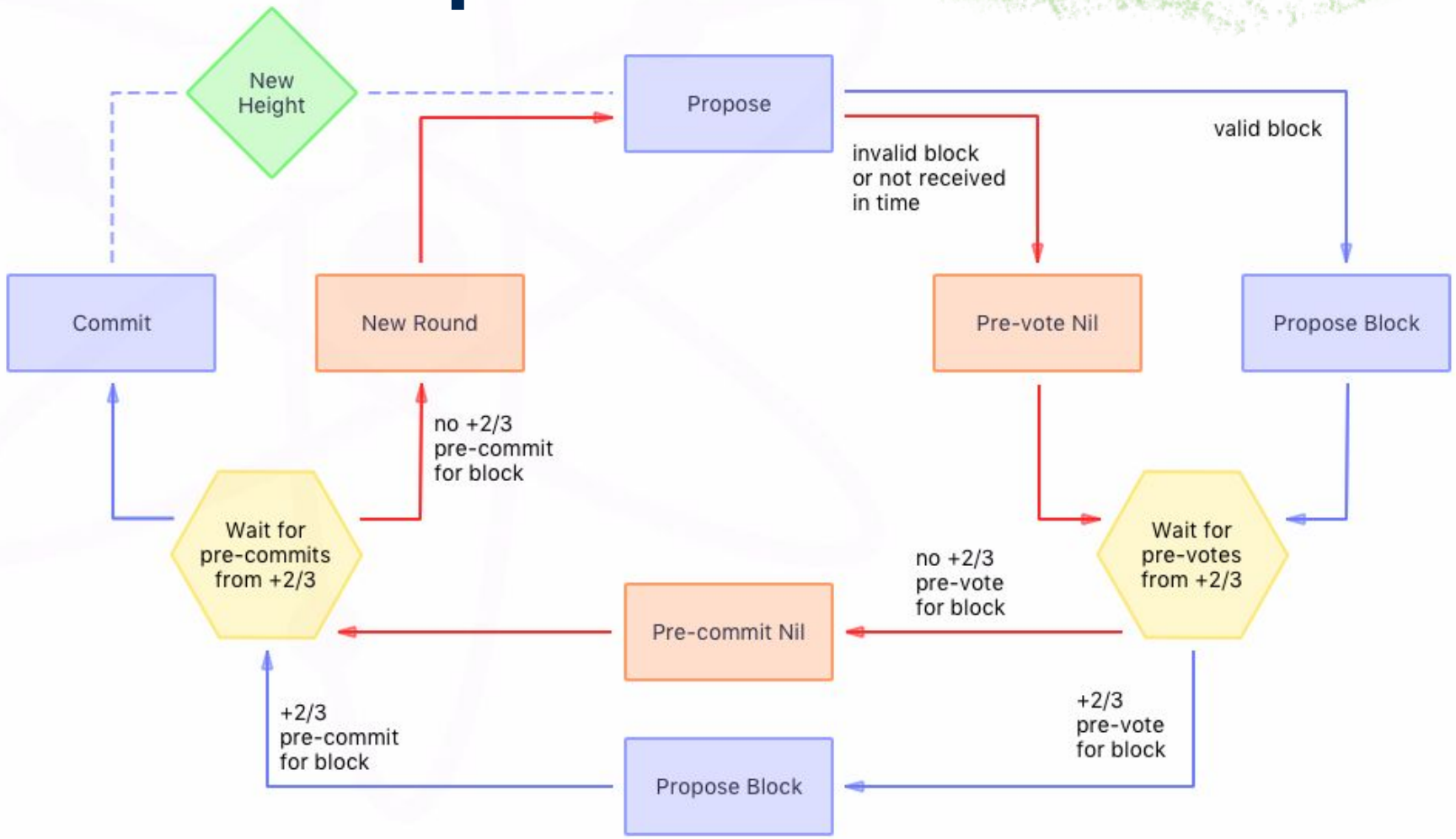
UCDAVIS COMPUTER SCIENCE

Source: PONTON: https://www.youtube.com/watch?v=AFiLxS7f1Lo

Advantages

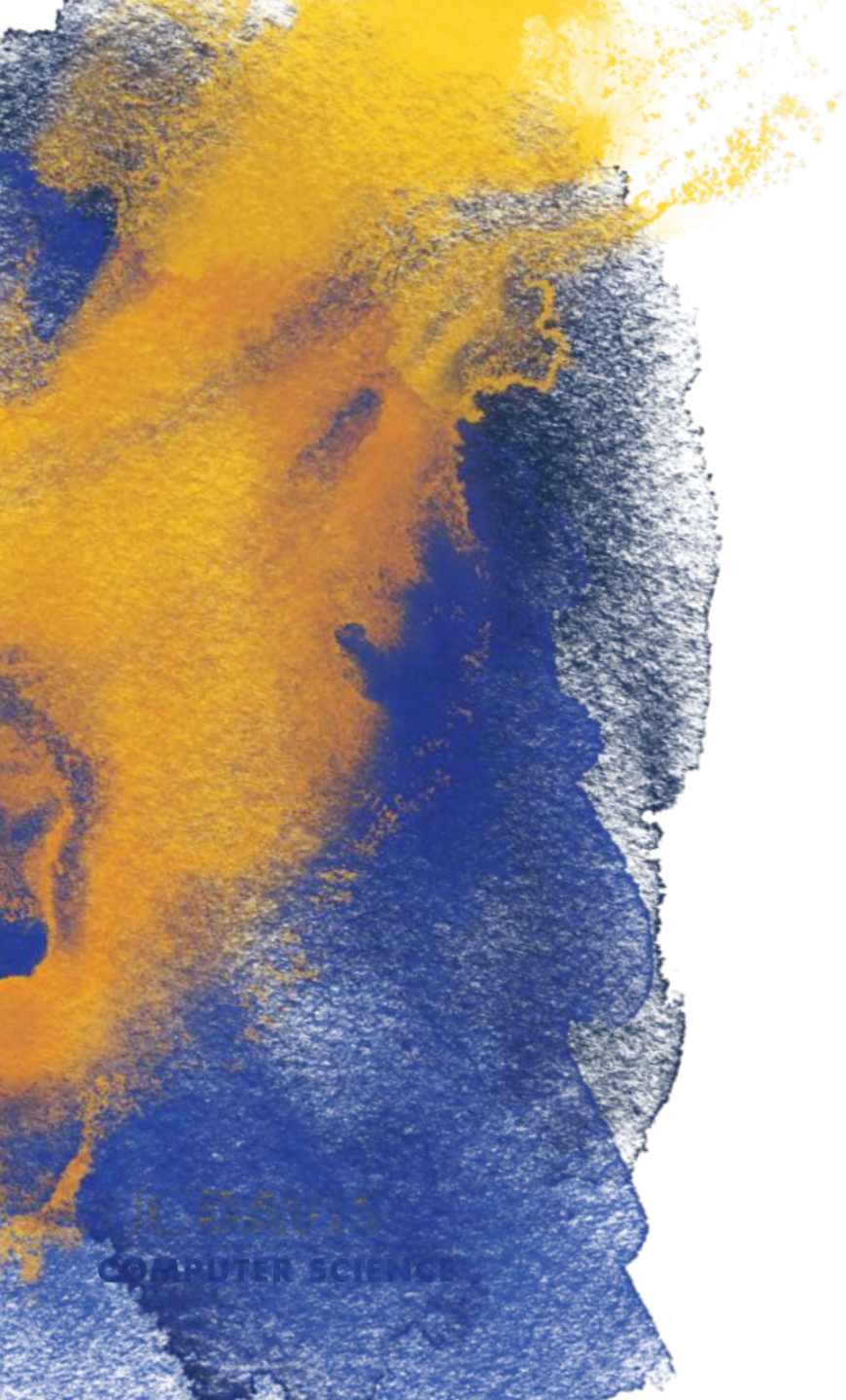
- **Simplification:** Tendermint is much simpler than PBFT. It removes the complex view change in PBFT when the leader changes.
- **Optimization for gossip protocols:** Traditional PBFT assumes point to point connections between all servers. Tendermint is optimized for P2P gossip protocols
- Tendermint is designed to process an entire block at a time.
- Tendermint is a professionally managed engine.

High-level Recap



UCDAVIS COMPUTER SCIENCE

Source: https://learn.bybit.com/blockchain/tendermint/



Thank you!

UCDAVIS
COMPUTER SCIENCE