Lecture 5: Consensus 101

Joachim Neu

https://jneu.net/

Important Disclaimers

The views expressed here are those of the academic researcher and not necessarily of AH Capital Management, L.L.C. (“a16z”).

This content is provided for academic and informational purposes only, and should not be relied upon as business, investment, tax or legal advice and does not constitute an offer of advisory services.

Charts, graphs or other diagrams provided within are for academic and informational purposes only and should not be relied upon when making any investment decision. Past performance is not indicative of future results. The content speaks only as of the date indicated.

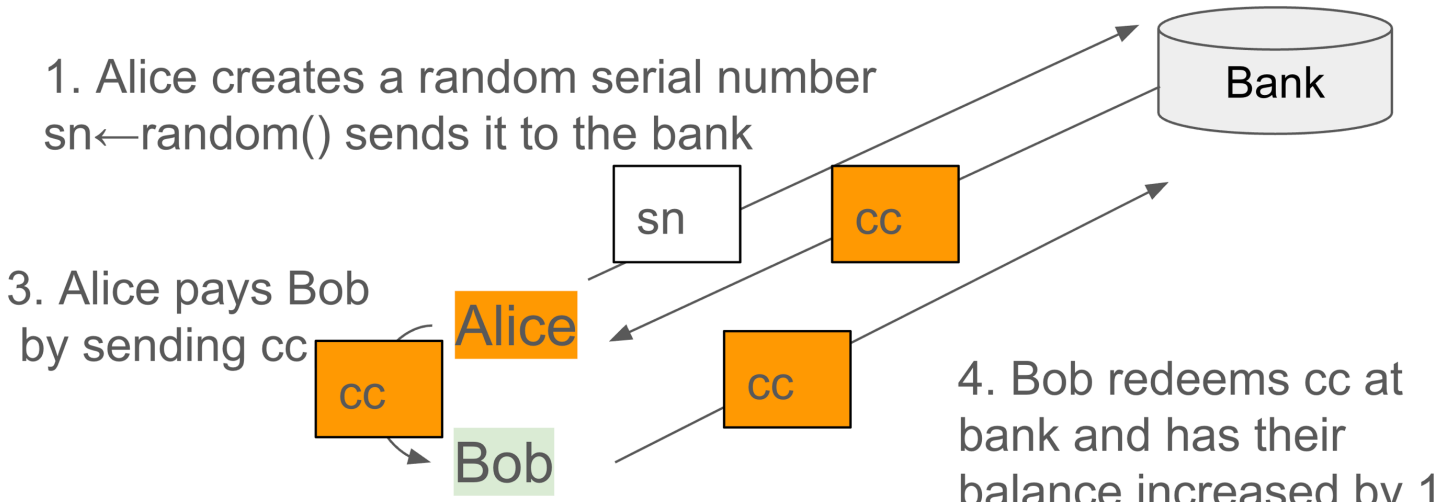
References to any securities, digital assets, and/or tokens (“assets”) are for illustrative purposes only and a16z may maintain holdings in these assets. A list of investments made by funds managed by Andreessen Horowitz (excluding investments for which the issuer has not provided permission for a16z to disclose publicly as well as unannounced investments in publicly traded digital assets) is available at https://a16z.com/investments/.

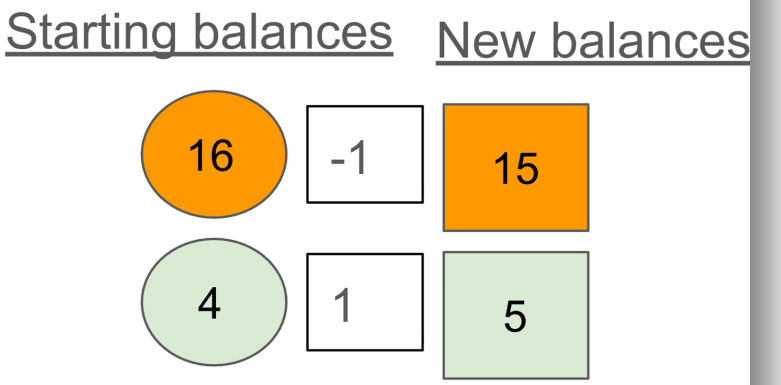
Recap

eCash (but with no privacy)

2. Bank signs sn to create a coin (cc)
Deducts 1 coin from Alice's balance

1. Alice creates a random serial number
sn←random() sends it to the bank

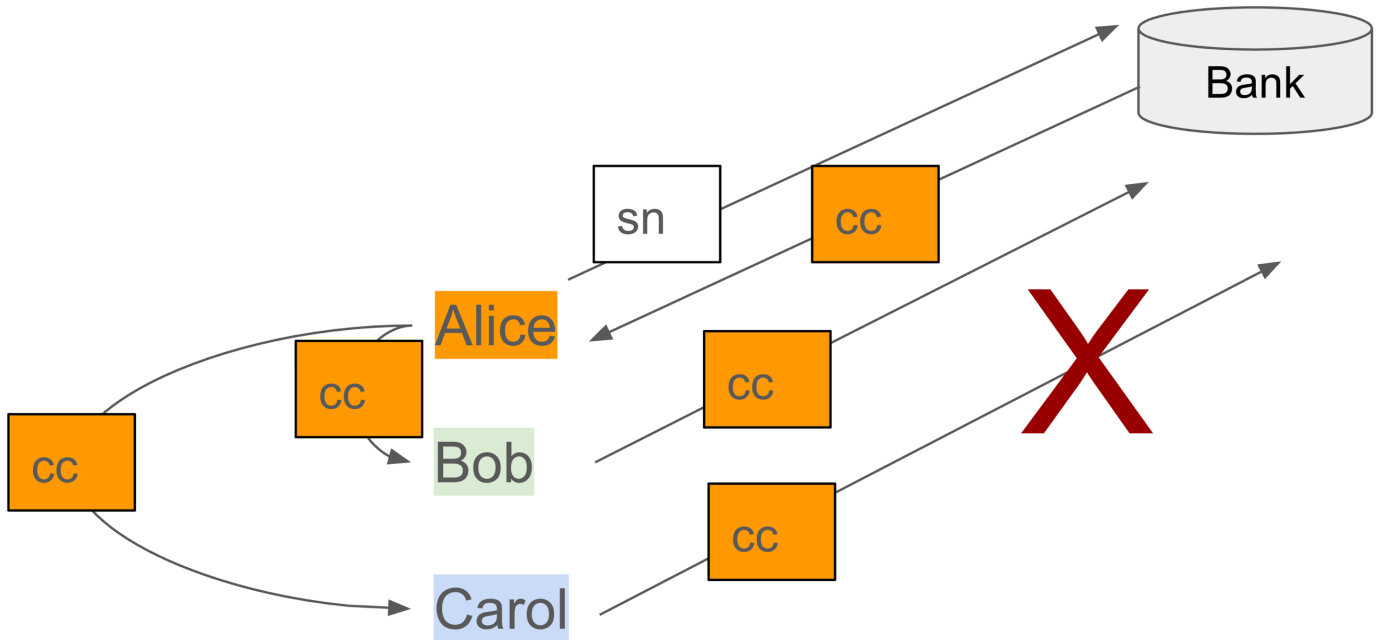




eCash bank issues and redeems coins

Recap

eCash (but with no privacy)

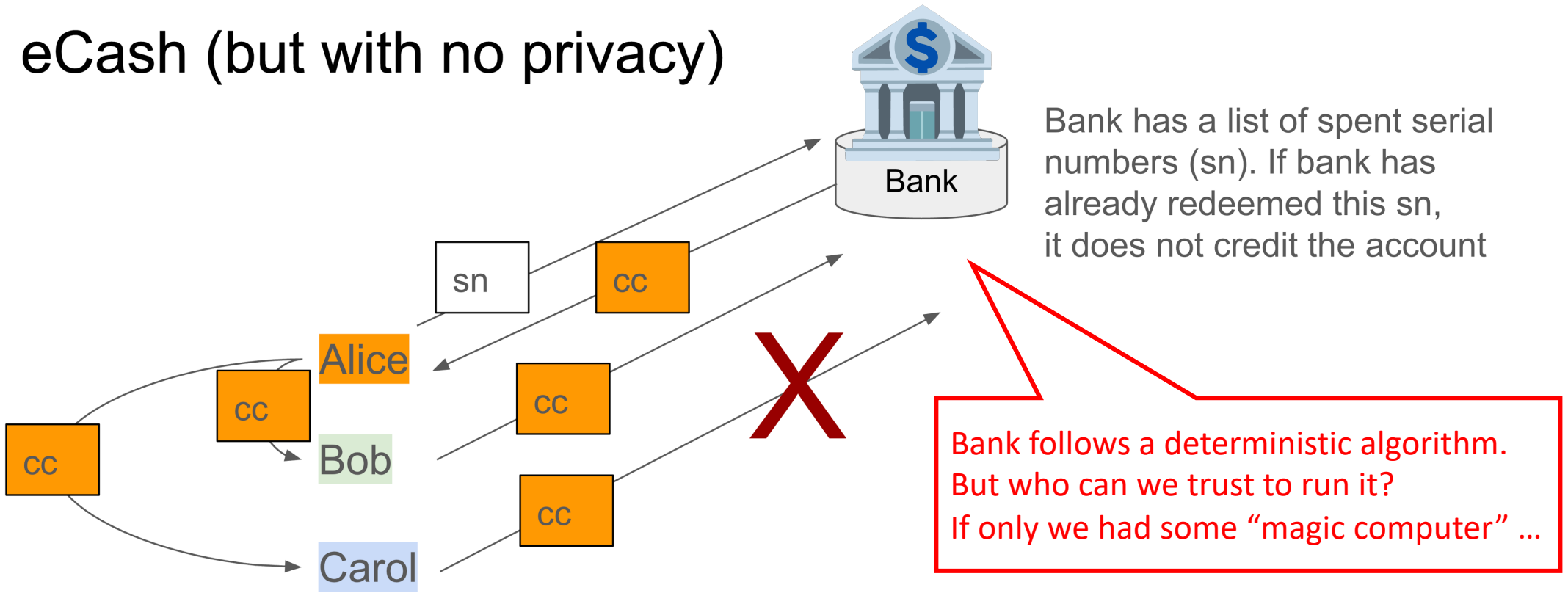


Bank has a list of spent serial numbers (sn). If bank has already redeemed this sn, it does not credit the account

What if Alice Doublespends?

Recap

eCash (but with no privacy)



Bank has a list of spent serial numbers (sn). If bank has already redeemed this sn, it does not credit the account

Bank follows a deterministic algorithm.
But who can we trust to run it?
If only we had some "magic computer" …

What if Alice Doublespends?

Magic Computer

Everybody can (using egalitarian interface) { **read** state and **provide** input

Nobody can { **stop** or **corrupt** } execution

Magic Computer

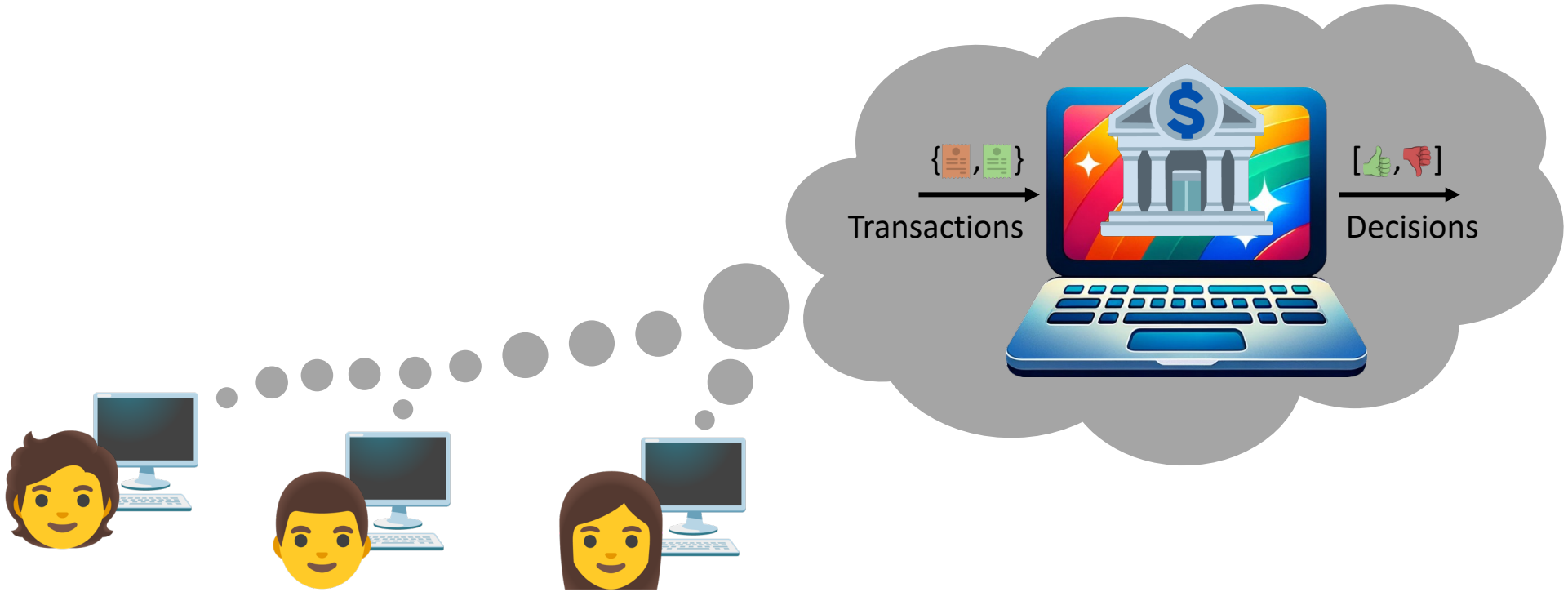
Everybody can (using egalitarian interface) { **read** state / and **provide** input }

Nobody can { **stop** / or **corrupt** } execution

Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** state / and **provide** input }

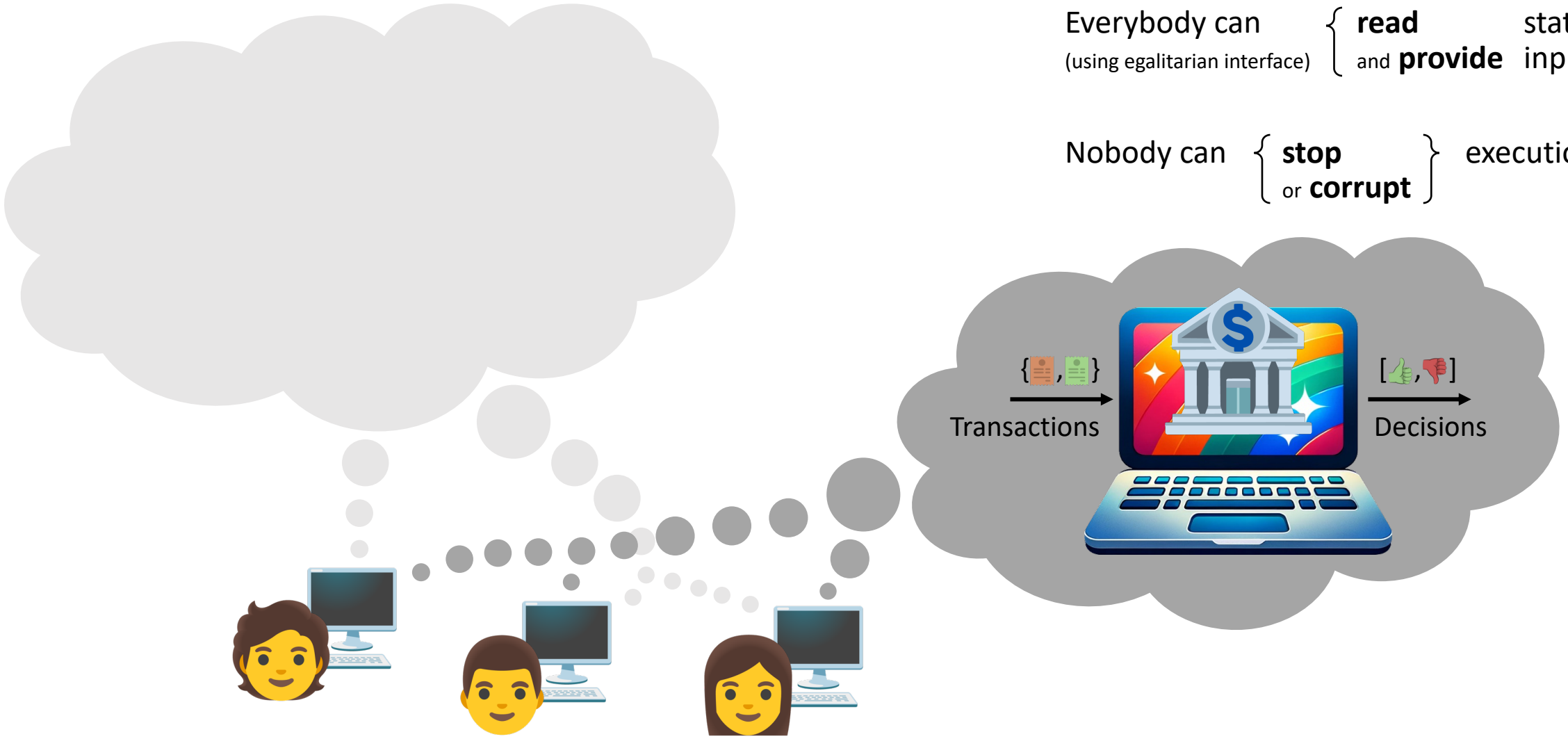
Nobody can { **stop** / or **corrupt** } execution



Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** state / and **provide** input }

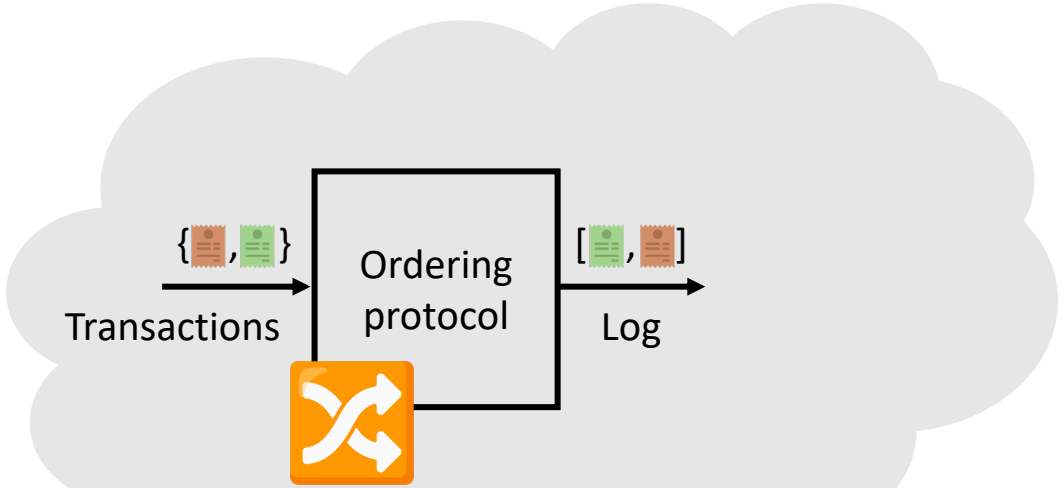
Nobody can { **stop** / or **corrupt** } execution

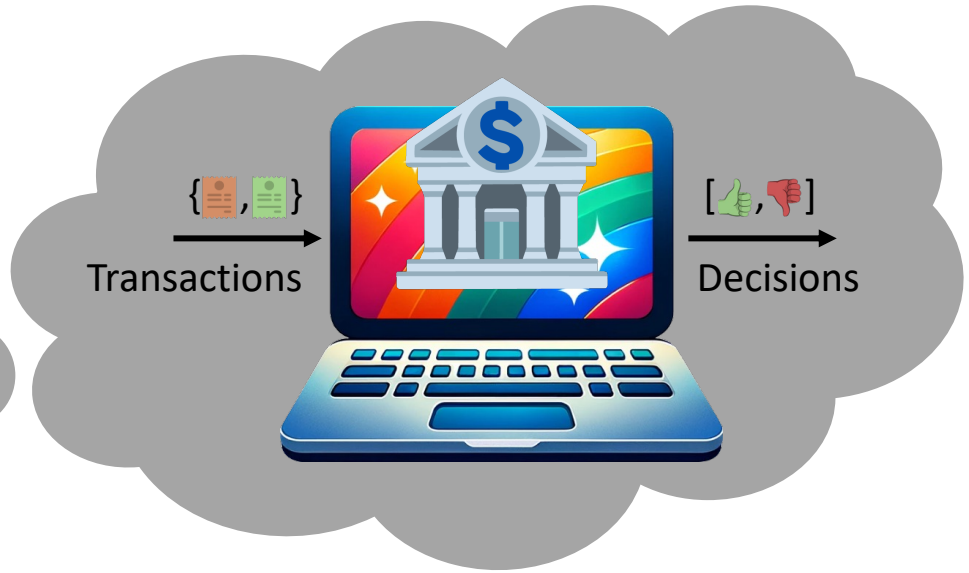


Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution

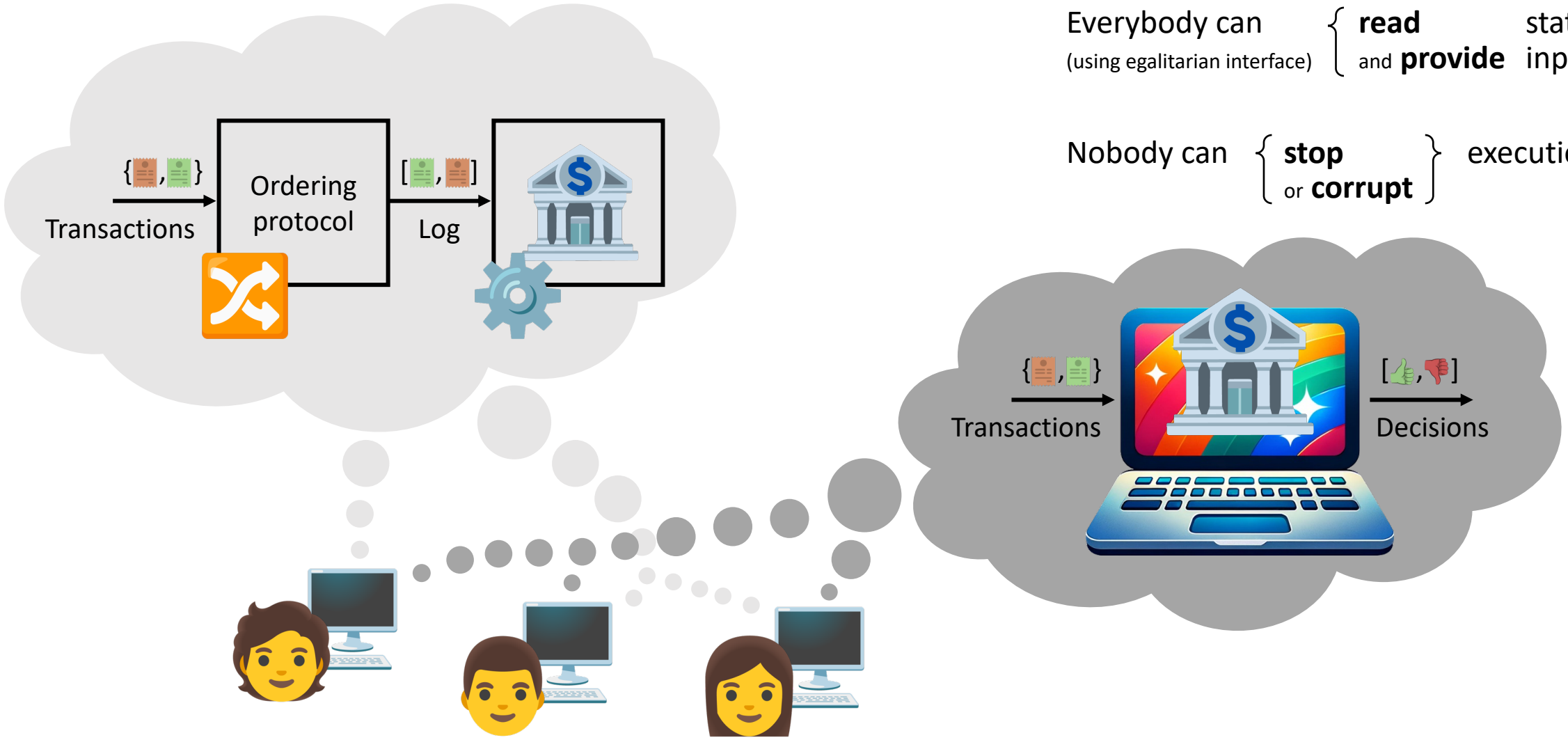




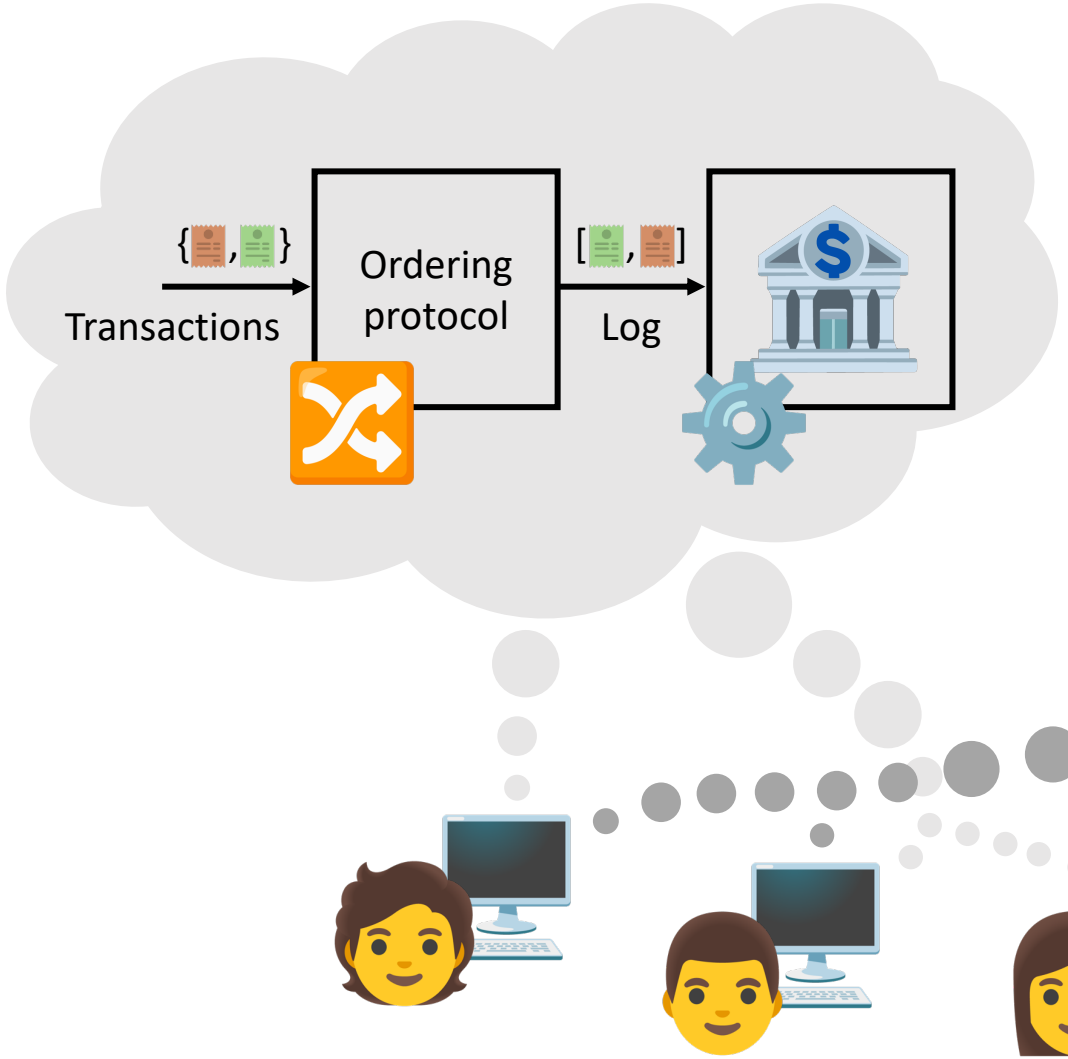
Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** state; and **provide** input }

Nobody can { **stop** or **corrupt** } execution



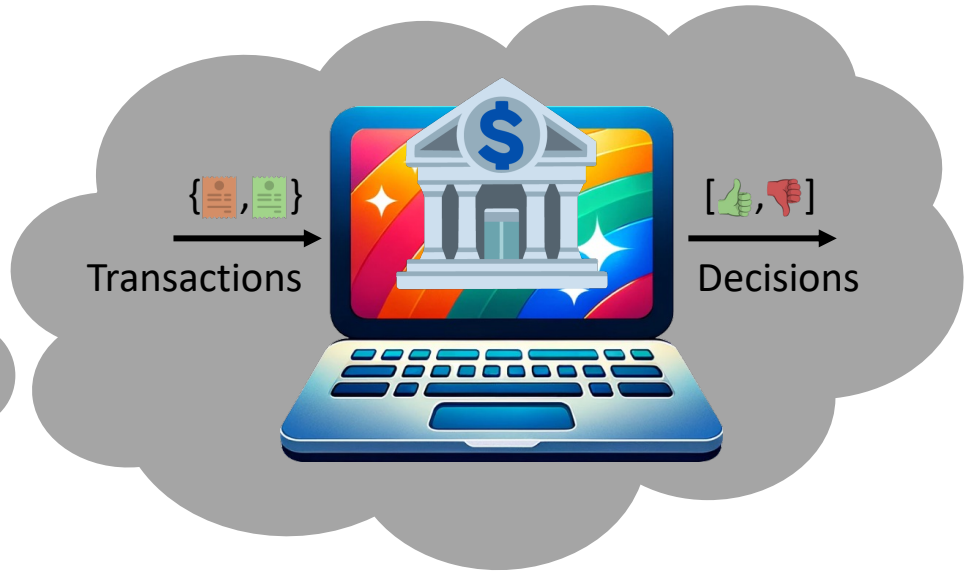
State-Machine Replication



Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution

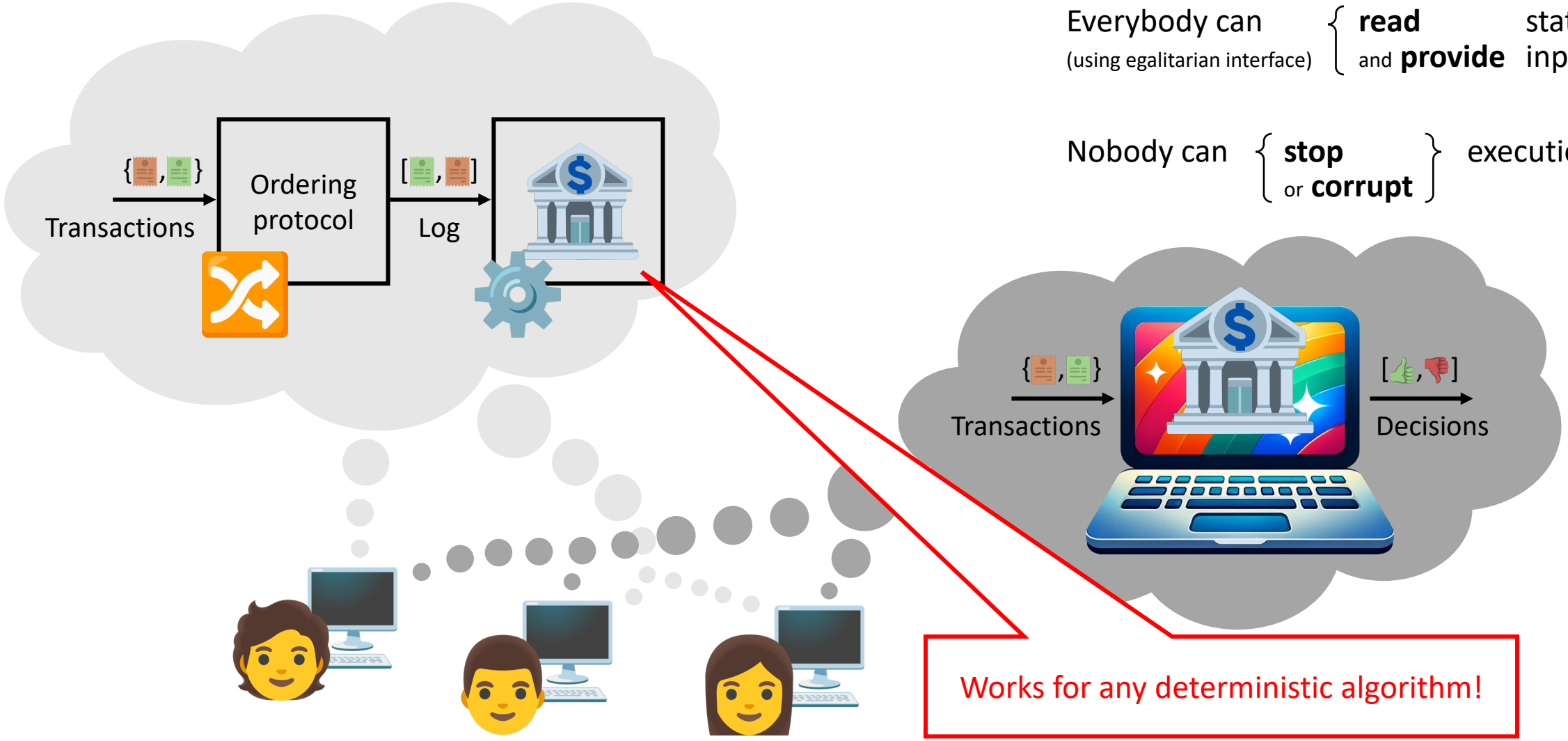


State-Machine Replication

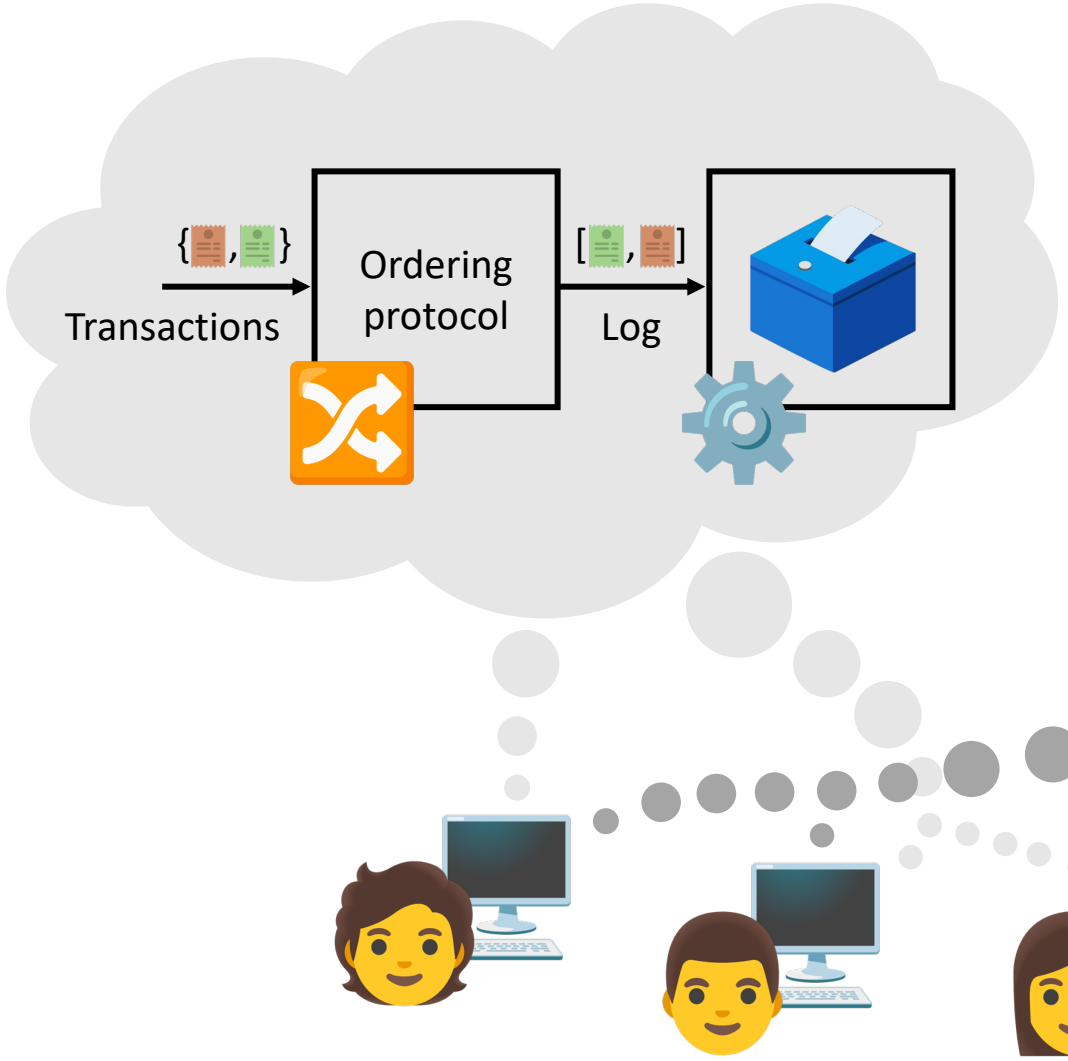
Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution



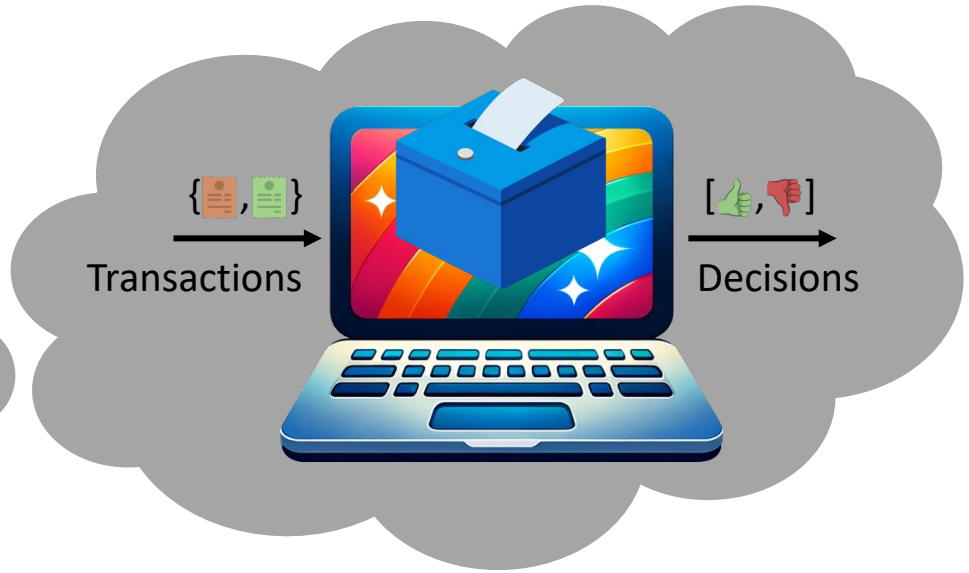
State-Machine Replication



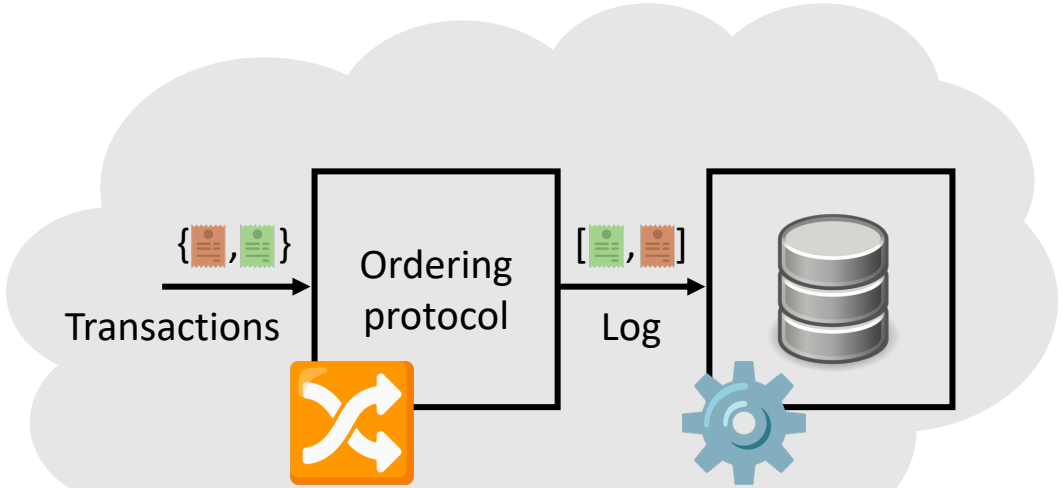
Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution



State-Machine Replication

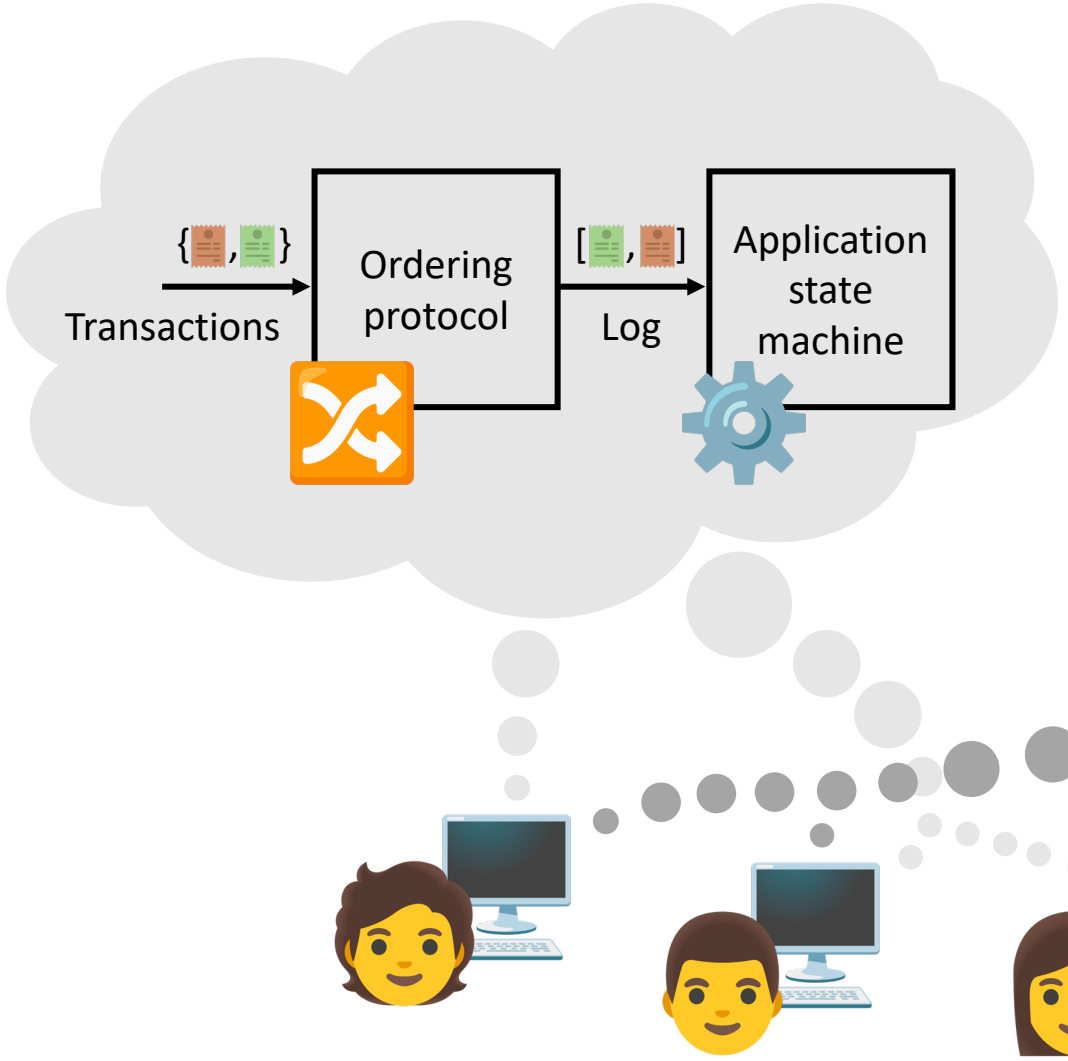


Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution

State-Machine Replication

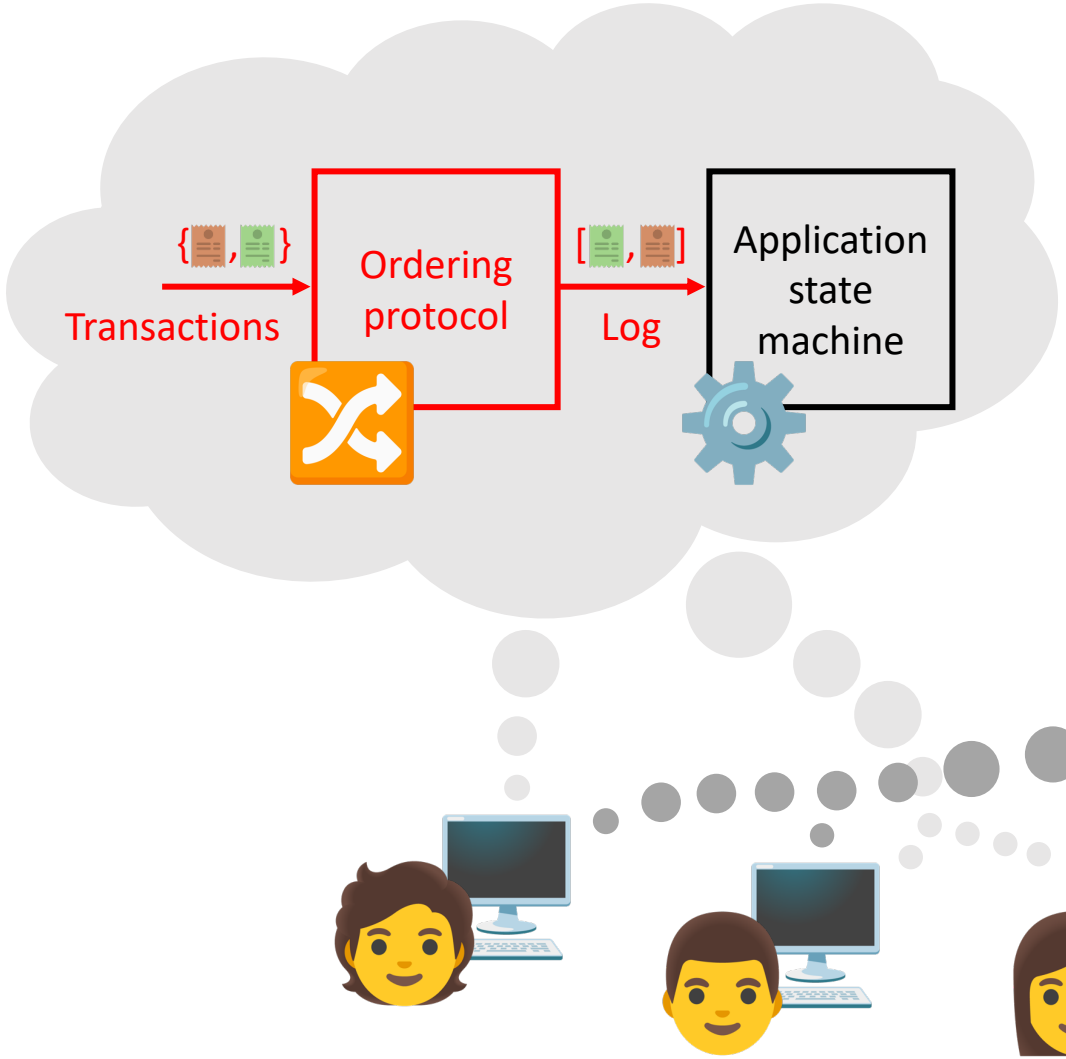


Magic Computer In The Sky

Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution

State-Machine Replication



Magic Computer In The Sky

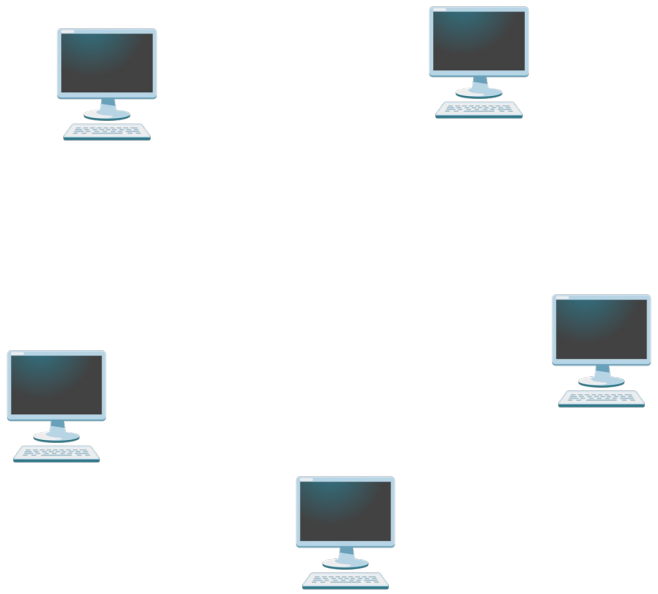
Everybody can (using egalitarian interface) { **read** and **provide** } state input

Nobody can { **stop** or **corrupt** } execution

State-Machine Replication (SMR) Consensus Protocol

State-Machine Replication (SMR): Interface

Nodes



State-Machine Replication (SMR): Interface

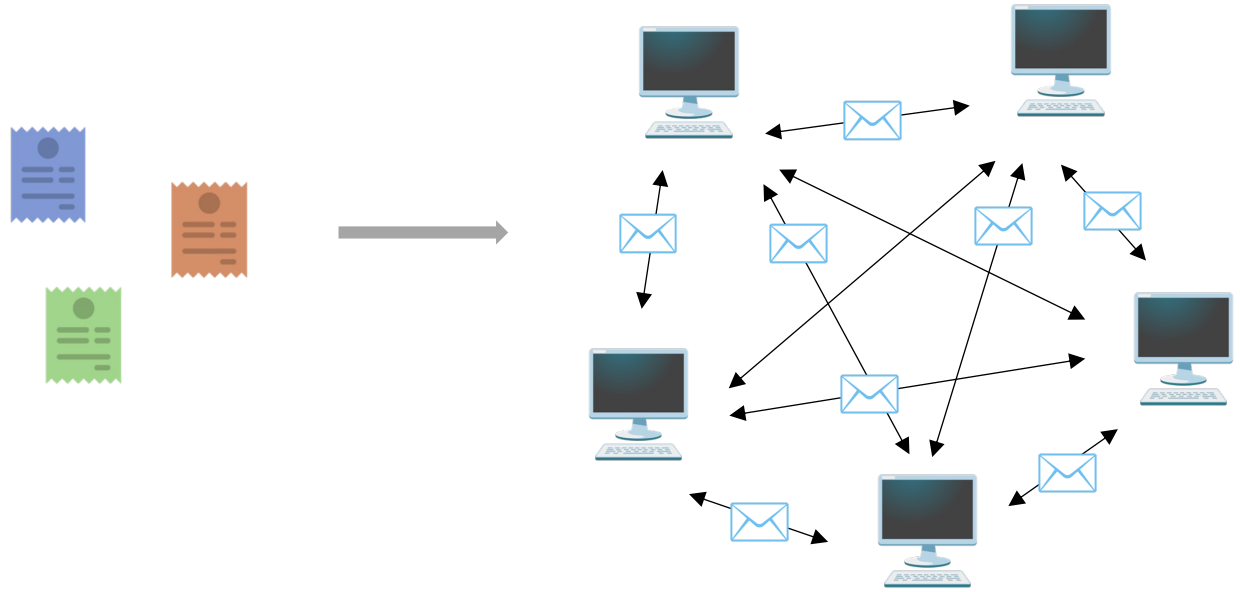
Input transactions

Nodes

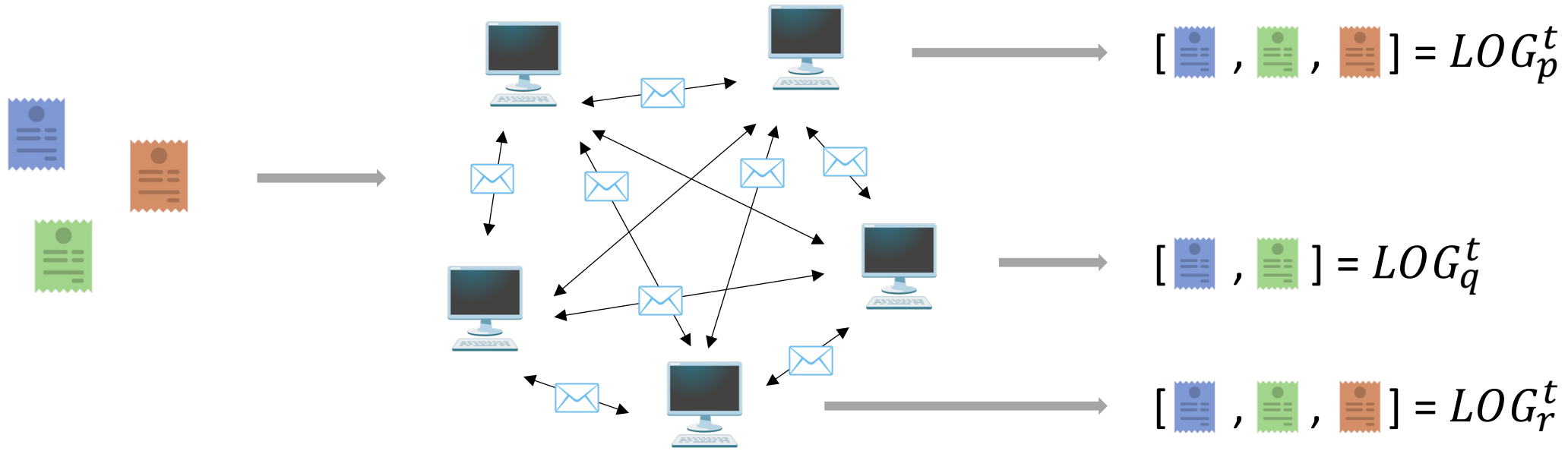
State-Machine Replication (SMR): Interface

Input transactions

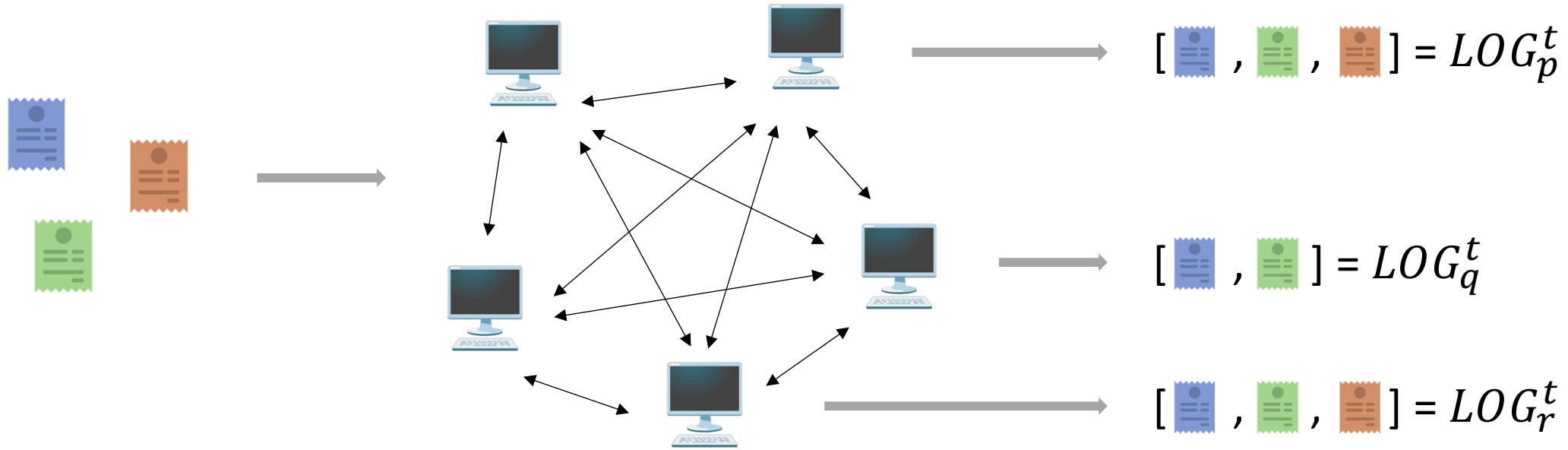
Nodes



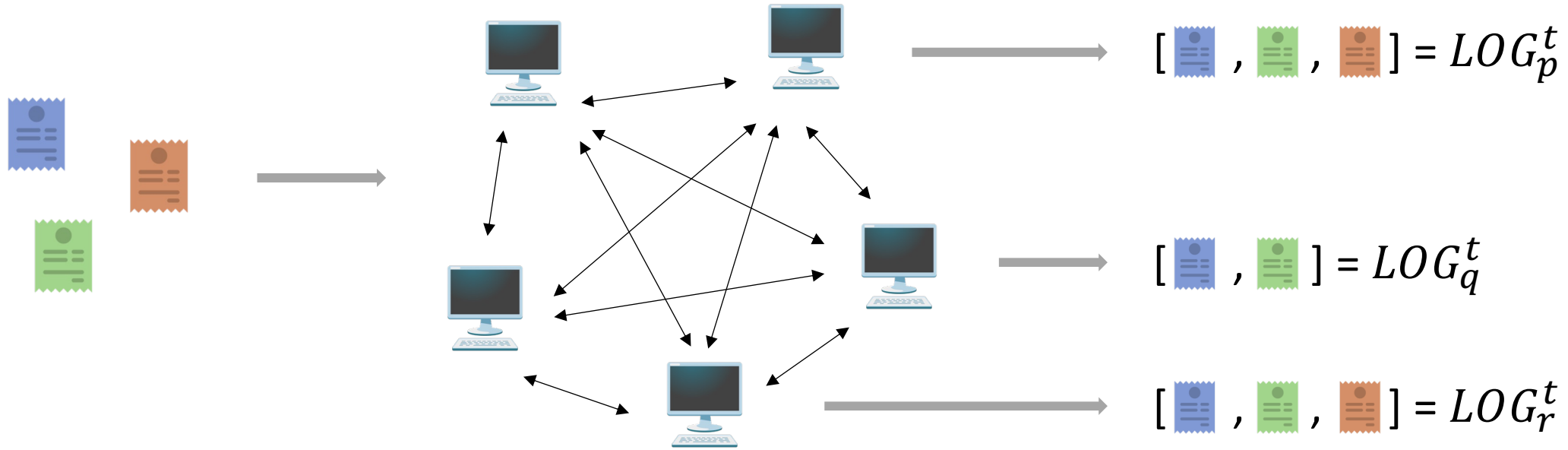
State-Machine Replication (SMR): Interface



State-Machine Replication (SMR): Interface

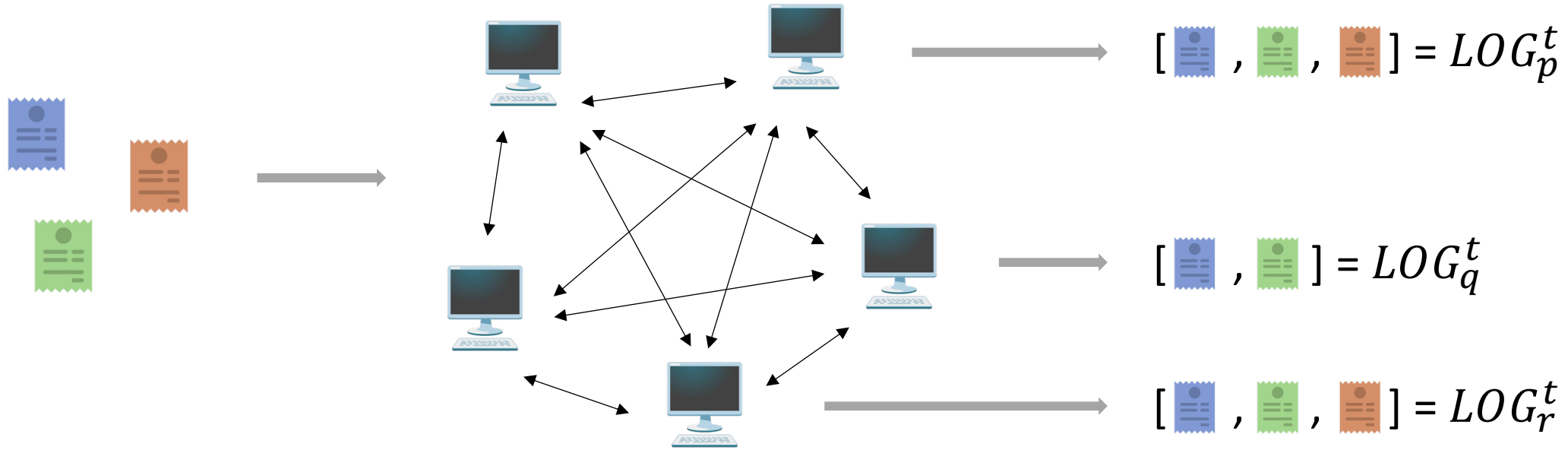


State-Machine Replication (SMR): Interface



Requirements

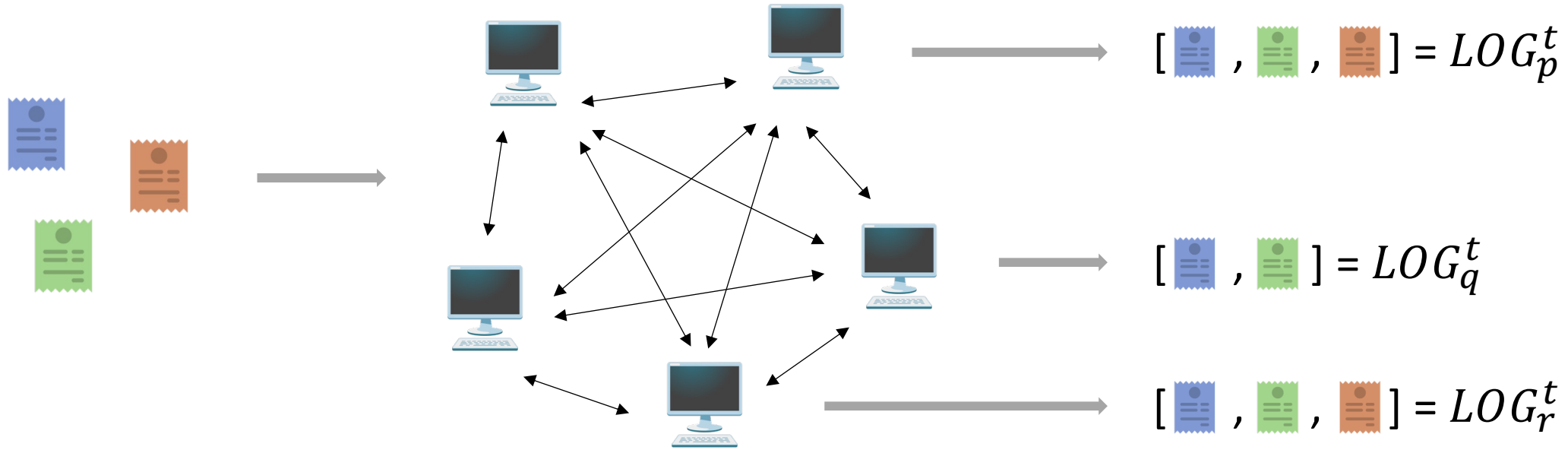
State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across nodes and times.

State-Machine Replication (SMR): Interface

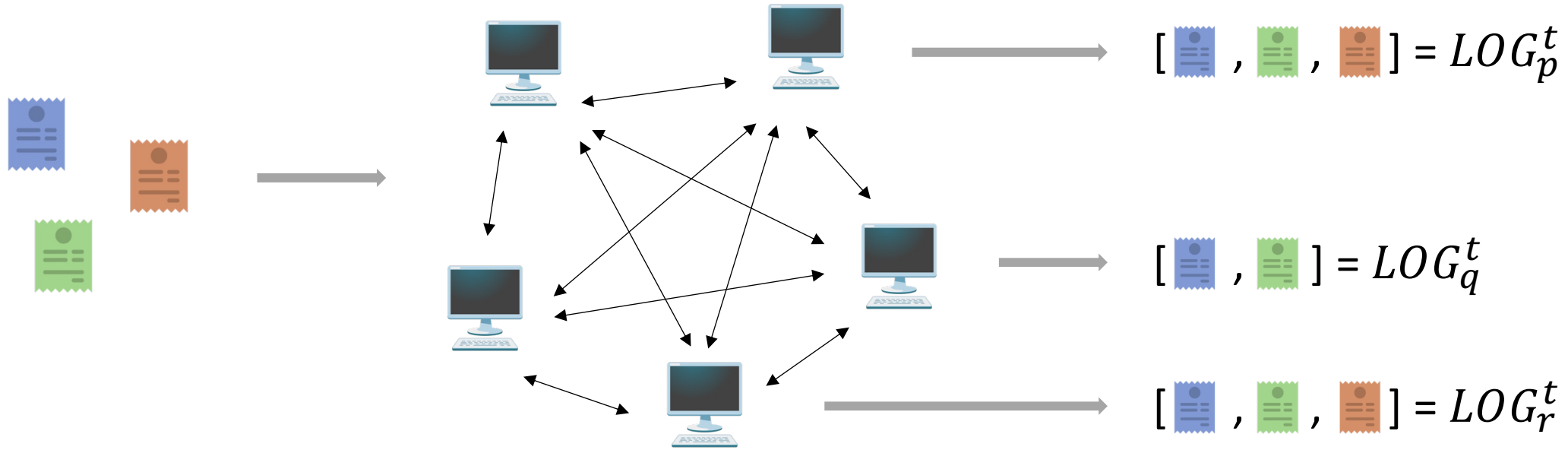


Requirements

Safety: Logs are consistent across nodes and times.

Bad things don't happen

State-Machine Replication (SMR): Interface



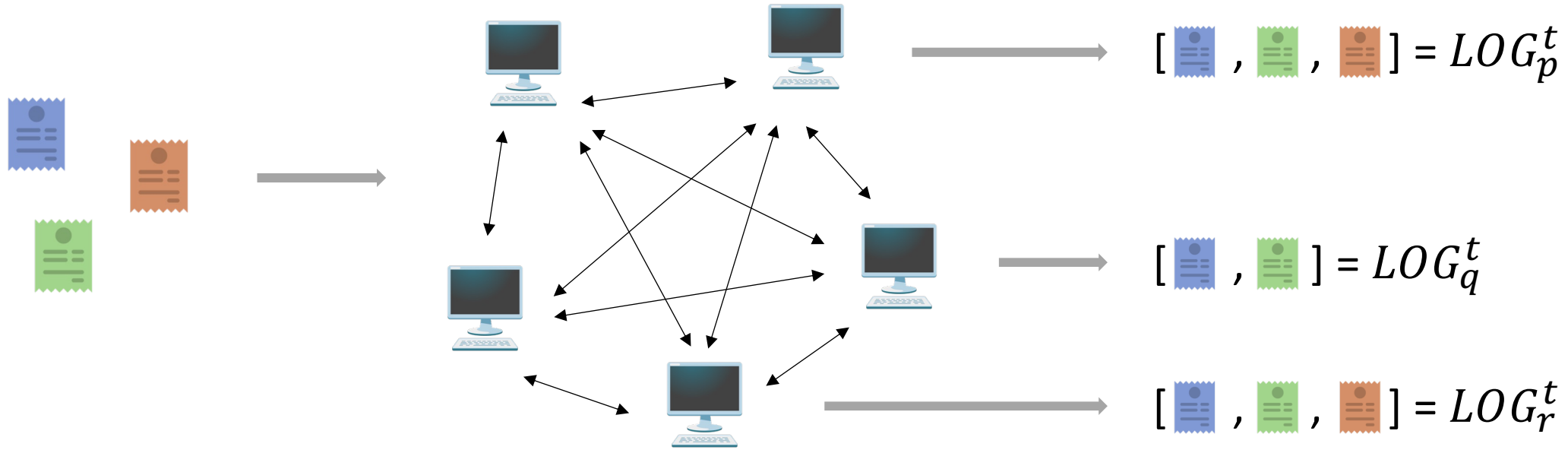
Requirements

Safety: Logs are consistent across nodes and times.

Bad things don't happen

Liveness: Transactions make it into every node's log eventually.

State-Machine Replication (SMR): Interface



Requirements

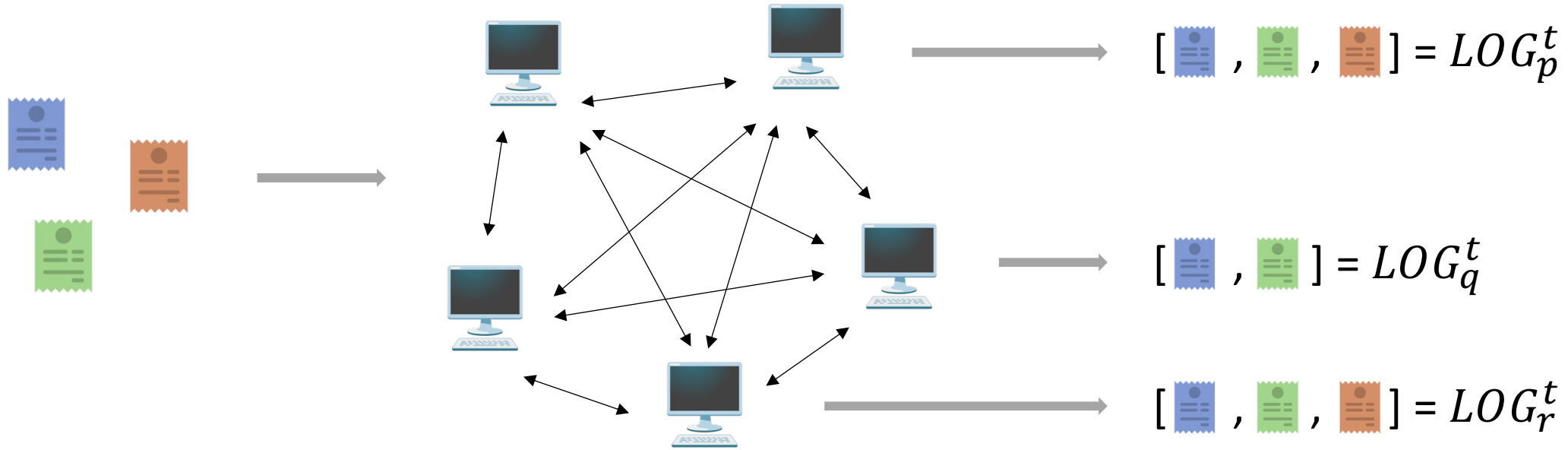
Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

Bad things don't happen

Good things happen

State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across nodes and times.

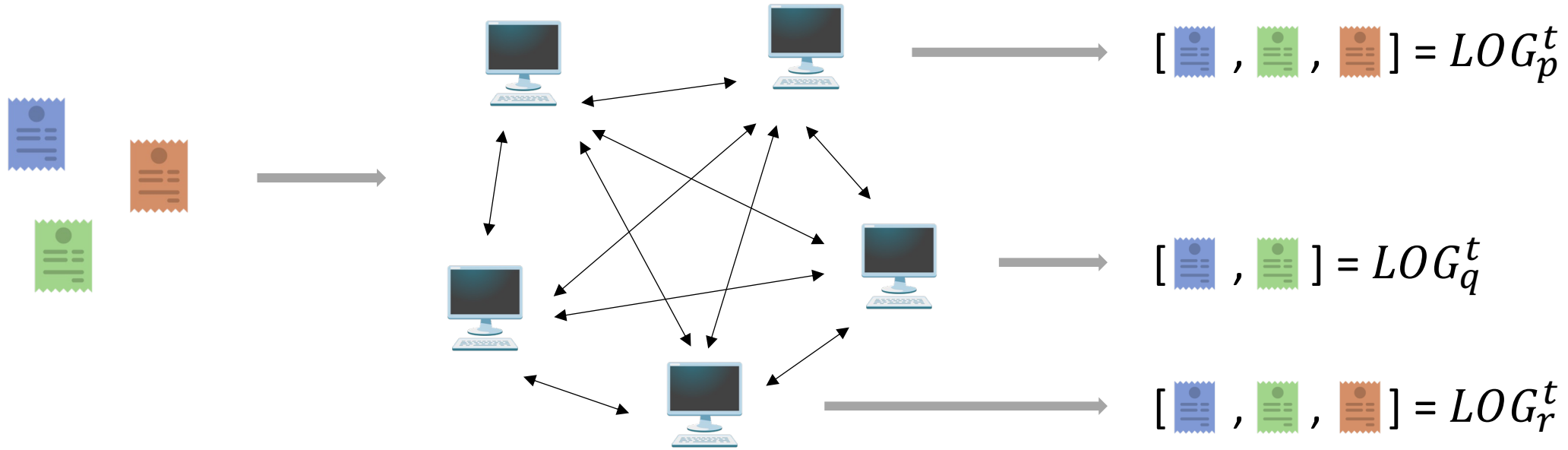
Liveness: Transactions make it into every node's log eventually.

Bad things don't happen

Good things happen

One without the other is trivial to achieve

State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

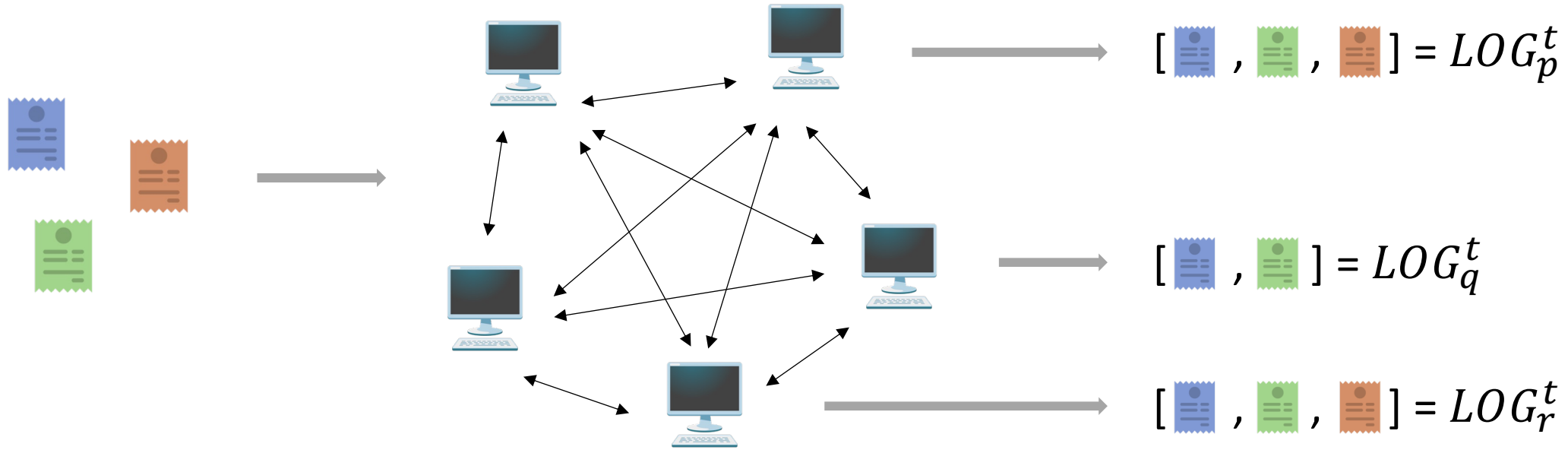
Bad things don't happen

Good things happen

One without the other is trivial to achieve

No ground truth ordering

State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

Bad things don't happen

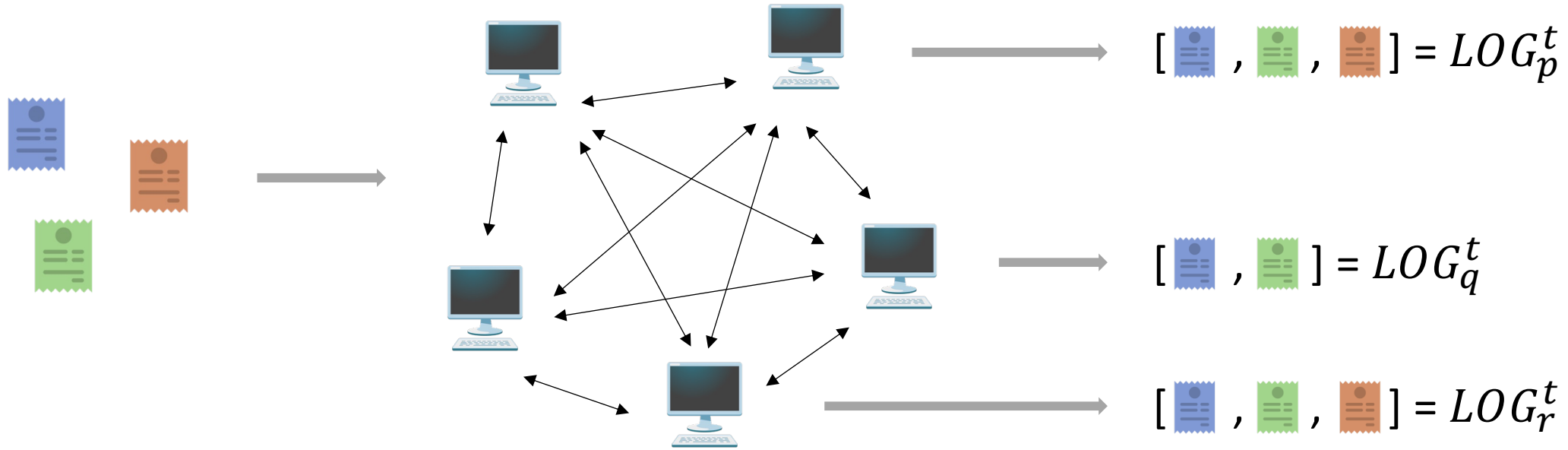
Good things happen

One without the other is trivial to achieve

No ground truth ordering

Transaction semantics not considered

State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

Bad things don't happen

Good things happen

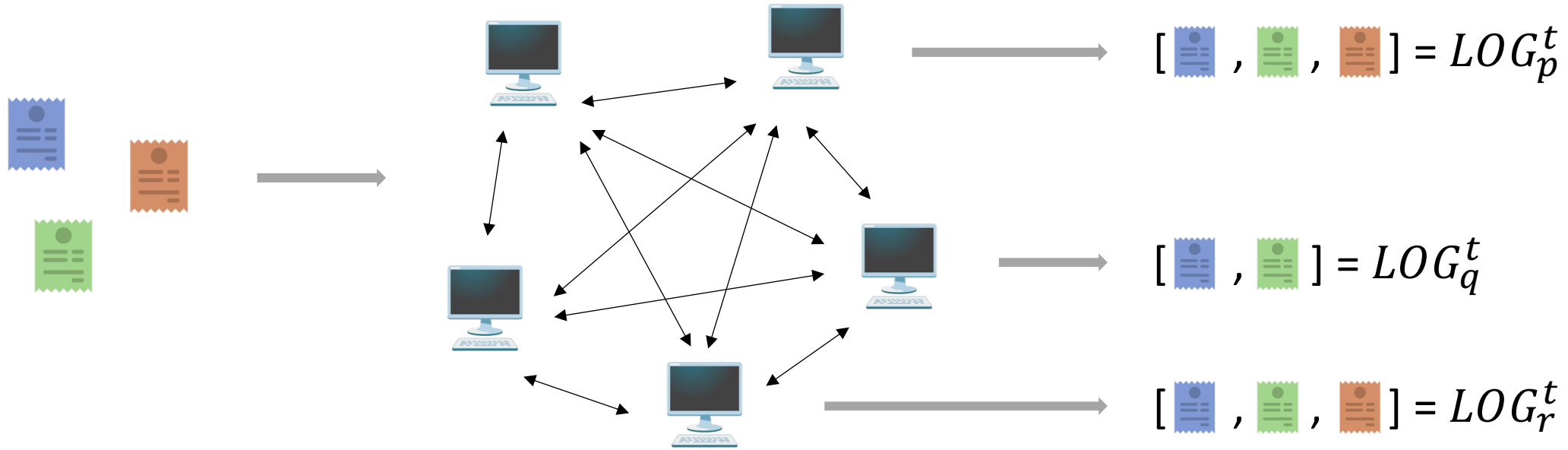
One without the other is trivial to achieve

No ground truth ordering

Transaction semantics not considered

Old computer science problem, studied since 1980s

State-Machine Replication (SMR): Interface

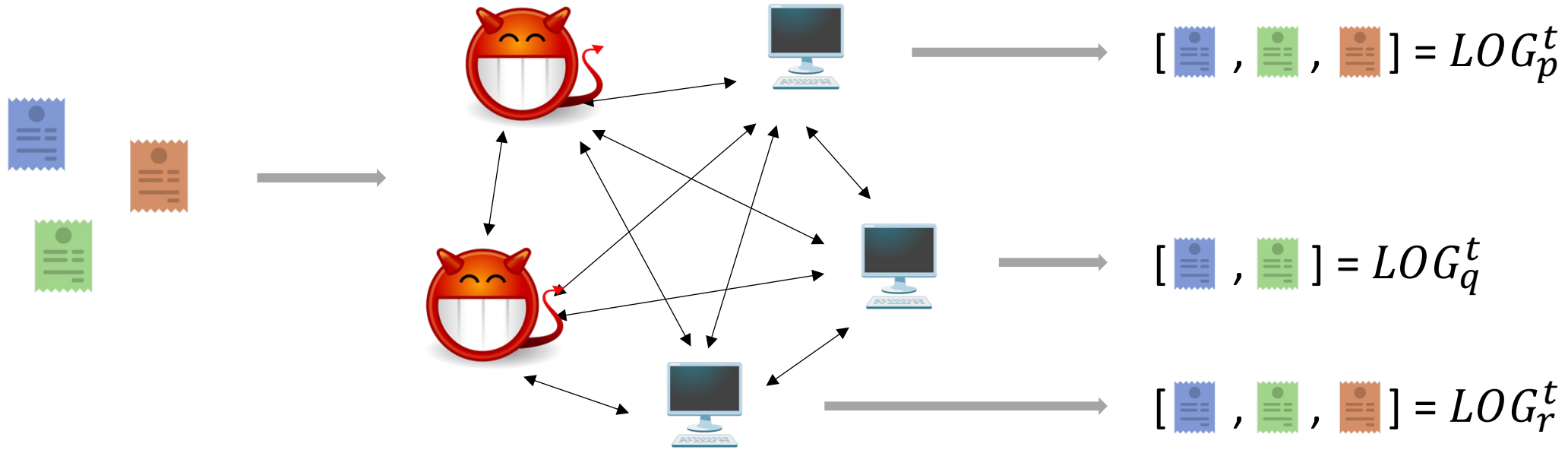


Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

State-Machine Replication (SMR): Interface

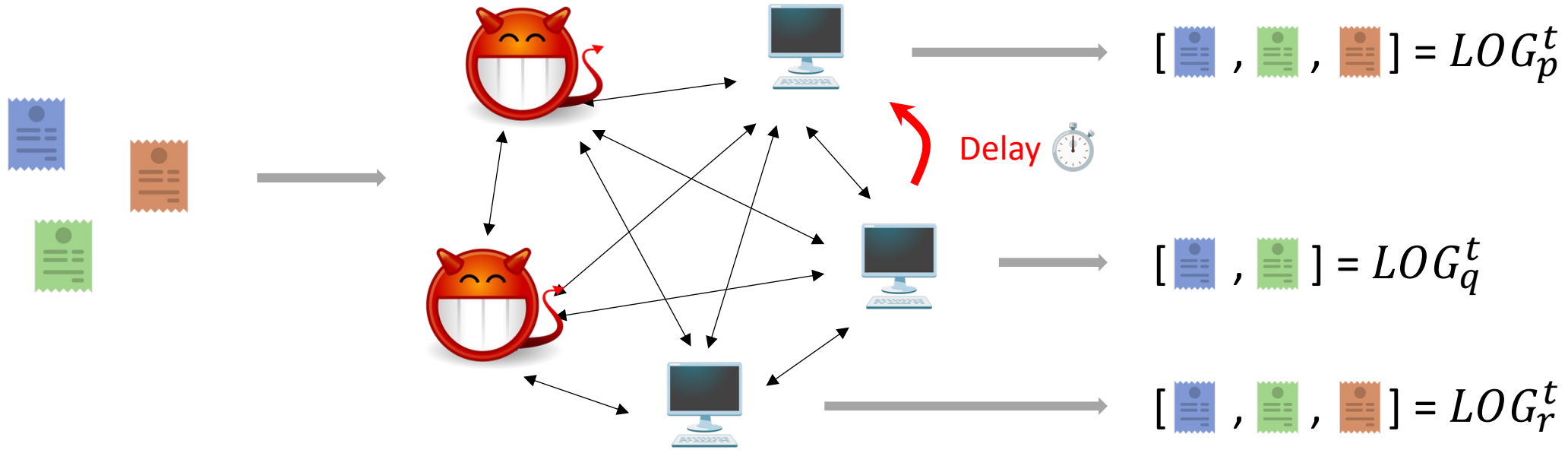


Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

State-Machine Replication (SMR): Interface

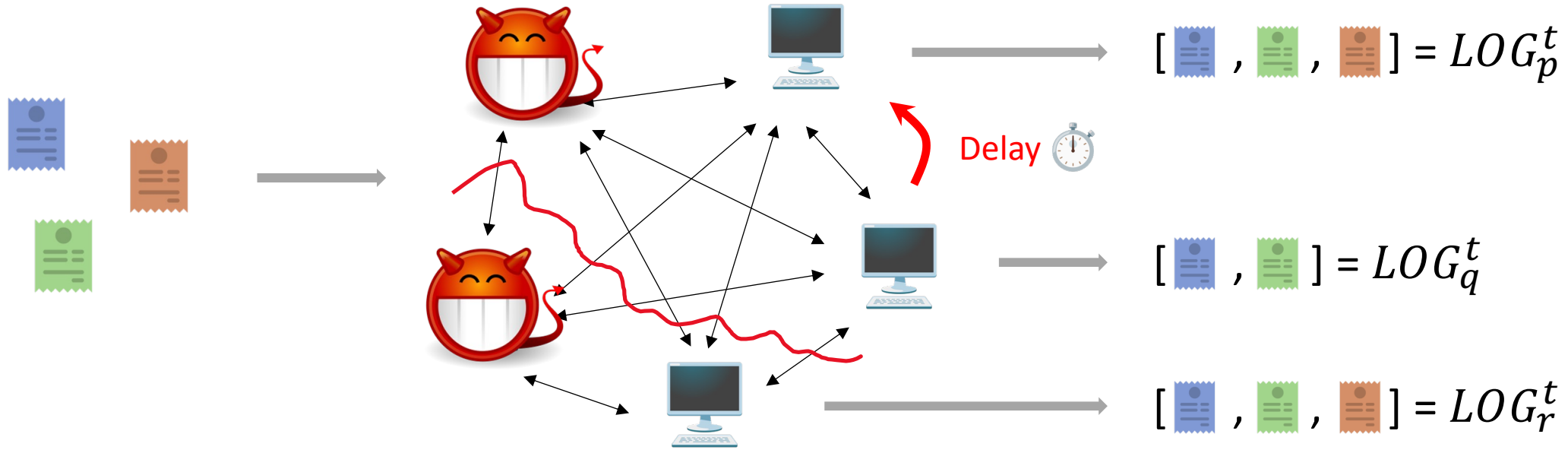


Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

State-Machine Replication (SMR): Interface

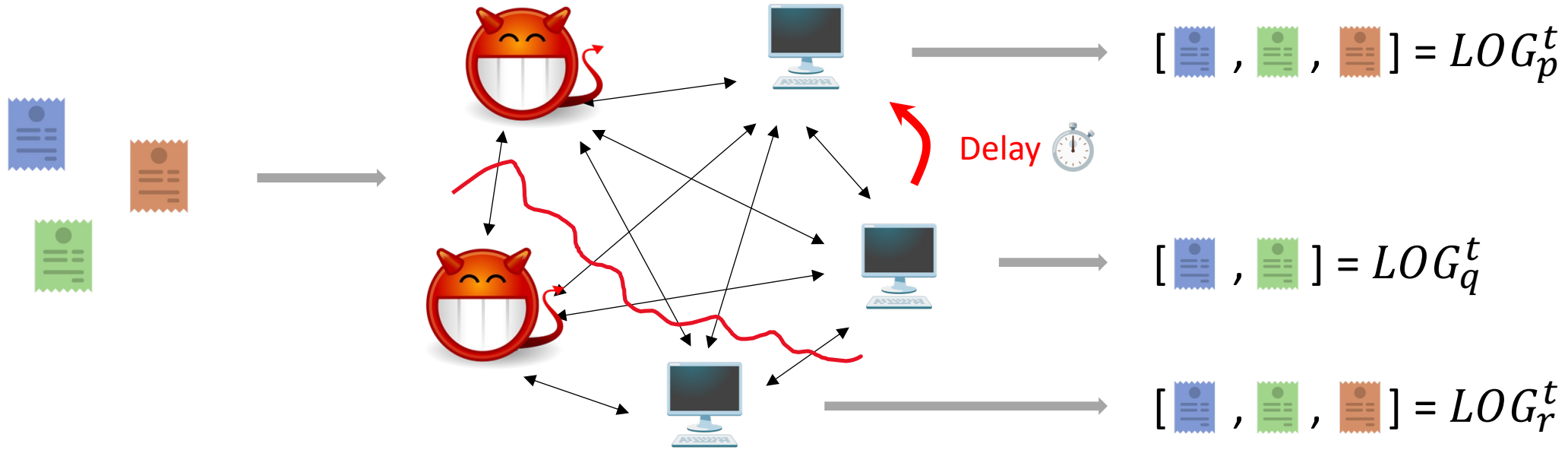


Requirements

Safety: Logs are consistent across nodes and times.

Liveness: Transactions make it into every node's log eventually.

State-Machine Replication (SMR): Interface

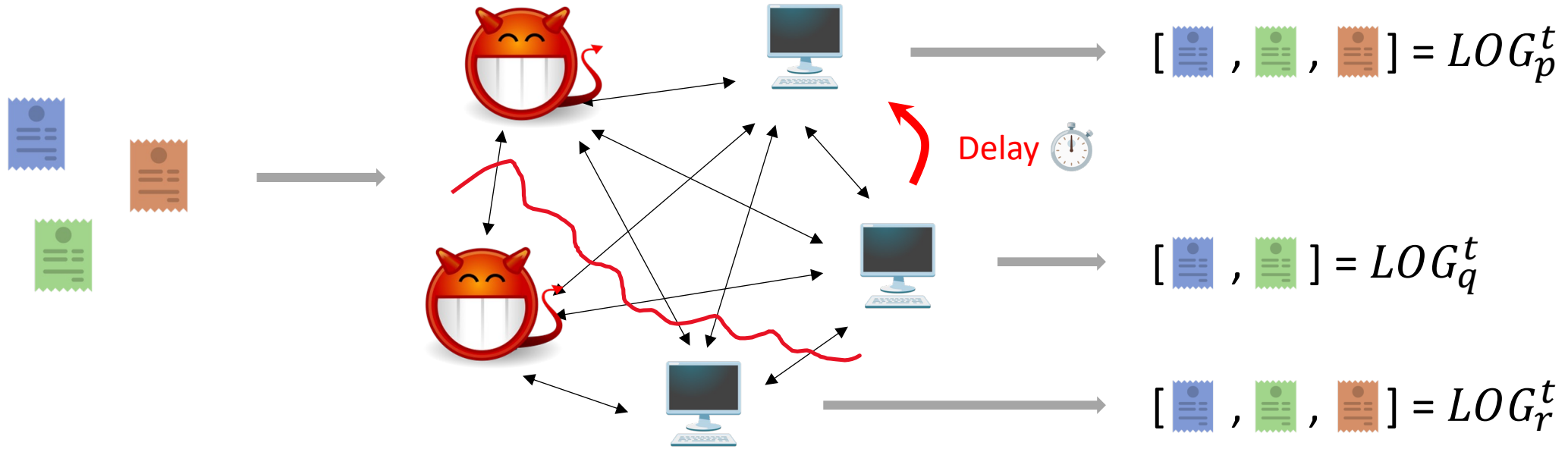


Requirements

Safety: Logs are consistent across honest nodes and times

Liveness: Transactions make it into every honest node's log eventually

State-Machine Replication (SMR): Interface

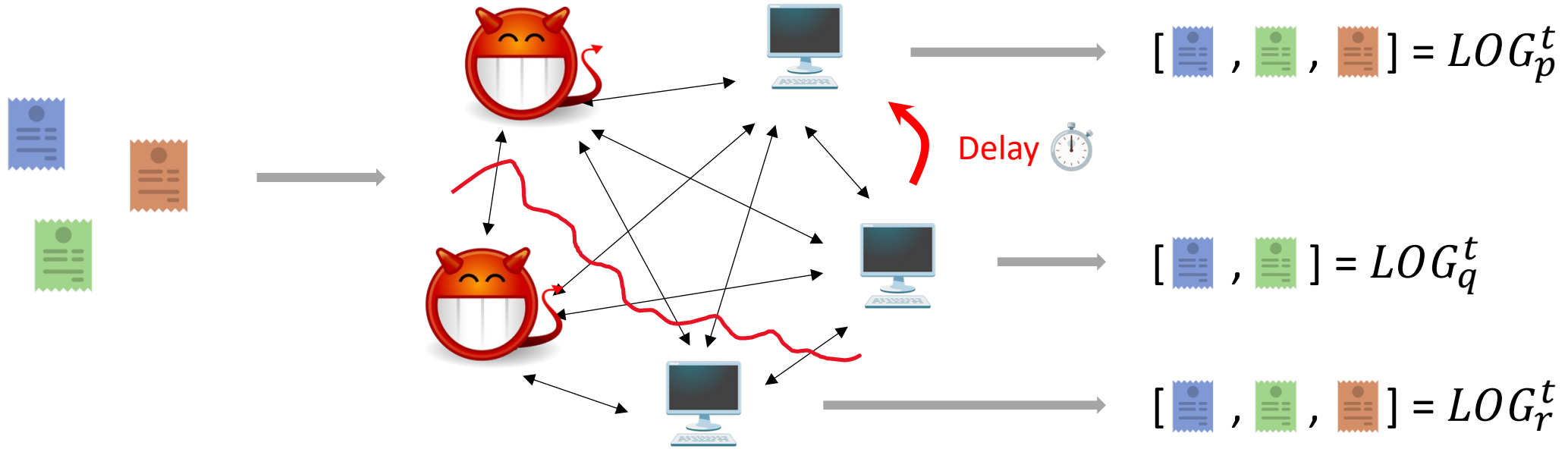


Requirements

Safety: Logs are consistent across honest nodes and times ... under conditions on 😈$\mathcal{S}$, ⏱️.

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈$\mathcal{S}$, ⏱️.

State-Machine Replication (SMR): Interface

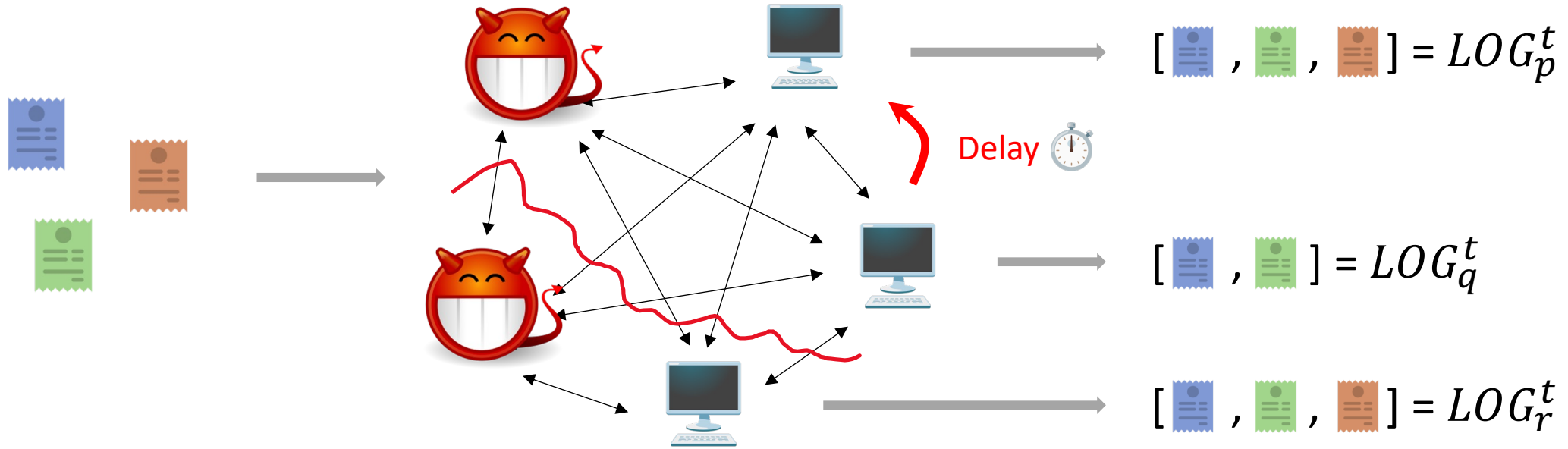


Requirements

Safety: Logs are consistent across honest nodes and times ... under conditions on 😈, ⏱️.

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈, ⏱️.

State-Machine Replication (SMR): Interface



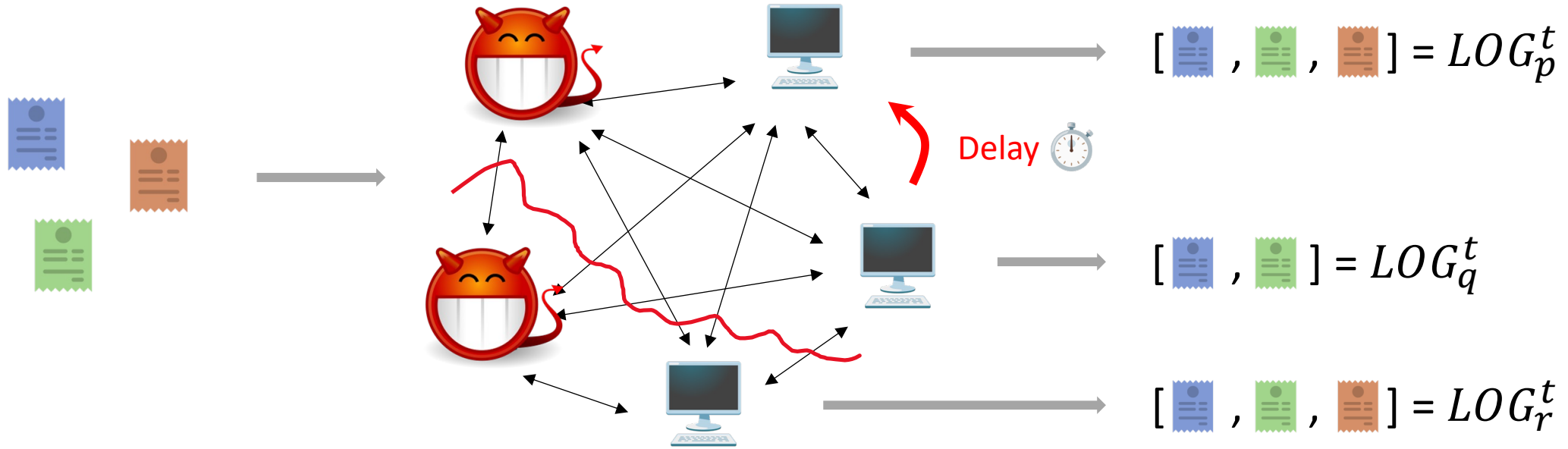
Requirements

Safety: Logs are consistent across honest nodes and times ... under conditions on 😈, ⏱️.

$$\forall \text{honest } p, p' \colon \quad \forall t, t' \colon \quad LOG_p^t \preccurlyeq LOG_{p'}^{t'} \ \vee \ LOG_{p'}^{t'} \preccurlyeq LOG_p^t$$

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈, ⏱️.

State-Machine Replication (SMR): Interface



Requirements

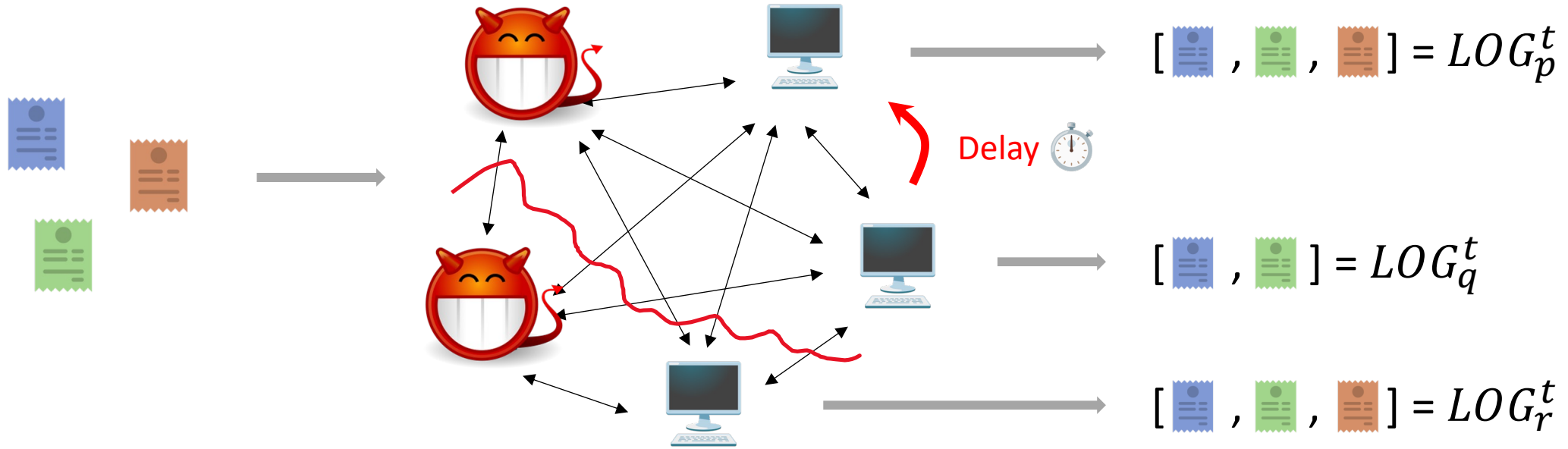
Safety: Logs are consistent across honest nodes and times ... under conditions on 😈, ⏱️.

$$\forall \text{honest } p, p' \colon \quad \forall t, t' \colon \quad LOG_p^t \preccurlyeq LOG_{p'}^{t'} \ \vee \ LOG_{p'}^{t'} \preccurlyeq LOG_p^t$$

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈, ⏱️.

$$\forall \text{input } tx \colon \quad \exists t_0 \colon \quad \forall t \geq t_0 \colon \quad \forall \text{honest } p \colon \quad tx \in LOG_p^t$$

State-Machine Replication (SMR): Interface

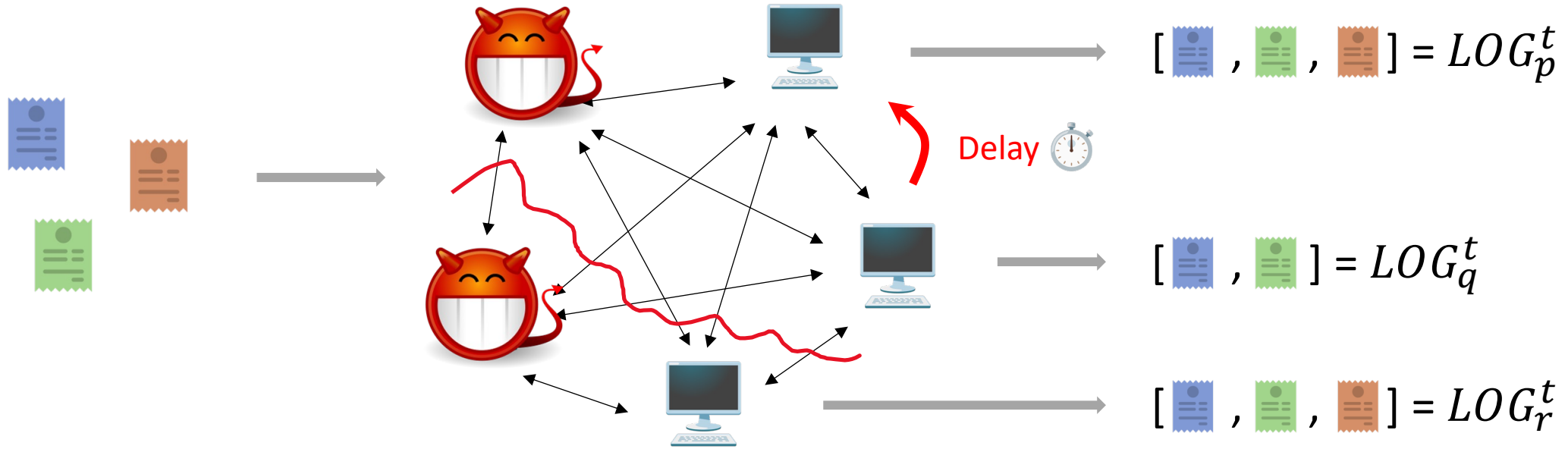


Requirements

Safety: Logs are consistent across honest nodes and times ... under conditions on 😈, ⏱️.

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈, ⏱️.

State-Machine Replication (SMR): Interface



Requirements

Safety: Logs are consistent across honest nodes and times ... under conditions on 😈, ⏱️.

Liveness: Transactions make it into every honest node's log eventually ... under conditions on 😈, ⏱️.

Many variants of the consensus problem, depending on:
(1) What constitutes the set of nodes? (2) Network delay model. (3) Behaviors of adversarial nodes.
We will see some examples later!

(1) Set of Nodes

What constitutes “one node”? How to stop the adversary from pretending to have an inflated number of nodes (“Sybil attack”)? → “Sybil resistance mechanism”

- **Mostly historic: “authenticated point-to-point channels”** (“spoken model”)
  - n nodes; nodes “just know” which node they received a message from
- **Proof-of-work**
  - “One CPU, one vote”
  - Computational puzzles; solutions are found in proportion to compute power
  - See example later: hash puzzles in Bitcoin

Proof-of-work in consensus: new with Bitcoin

- **Permissioned** (cryptographic identities, “public key infrastructure”, “written model”)
  - n nodes, each with a secret/public key pair for a signature scheme
  - List of public keys is known to all nodes

“Permissioned consensus” ≠ “permissioned blockchains”

- **Proof-of-stake**
  - “One coin, one vote”
  - See example later: usually bootstrapped from permissioned setting

- **Proof-of-space, proof-of-useful-work/-space, …**

(2) Network Models

- **Synchronous network**

  There exists a delay upper-bound Δ, *known* to the nodes, so that every message sent between honest nodes at time t arrives by time $t + \Delta$.

Example protocols: Bitcoin, Cardano's Ouroboros, parts of Ethereum (LMD GHOST)

- **Partially synchronous network**

  There exists a delay upper-bound Δ, *known* to the nodes, *and an adversary-chosen "global stabilization time" GST*, so that every message sent between honest nodes at time t arrives by time $\max(GST, t) + \Delta$.

Example protocols: PBFT, Simplex, Tendermint, HotStuff, parts of Ethereum (Casper FFG)

- **Asynchronous network**

  Every message sent between honest nodes arrives *eventually*.

(3) Behaviors of Adversarial Nodes

- **Crash faults**

  Adversarial nodes stop protocol execution at some adversary-chosen time

- **Worst-case ("Byzantine") faults**

  Adversarial nodes follow adversary-chosen, computationally bounded behavior (→ cannot break cryptography), possibly coordinated across adversarial nodes

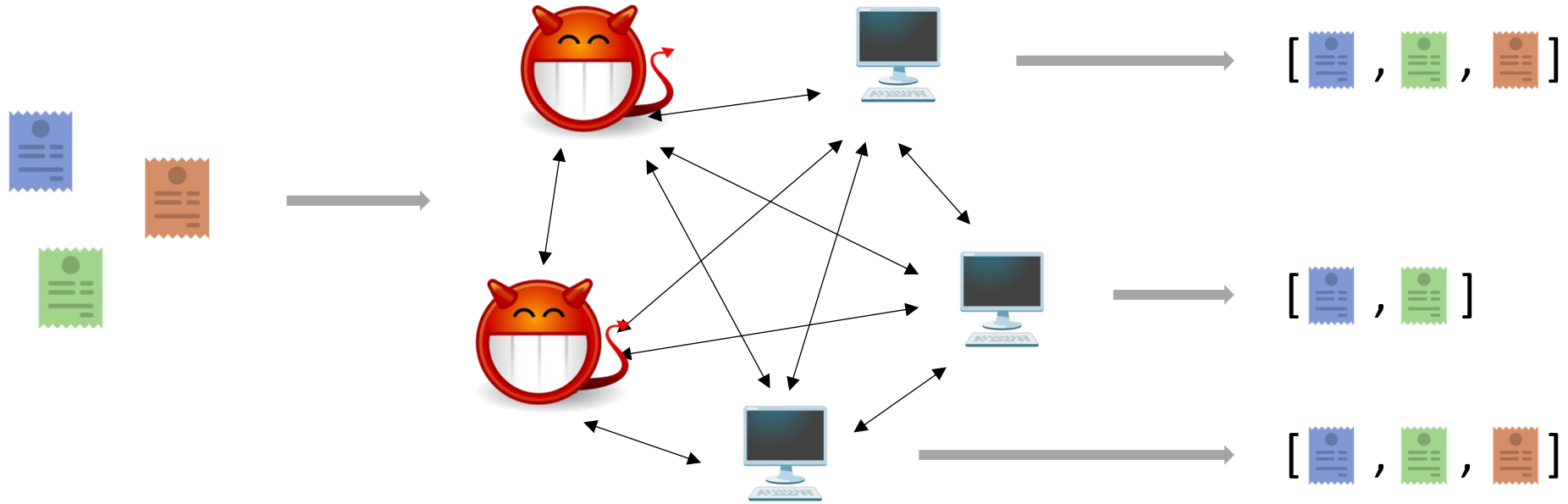
Model of choice for blockchains

- ~~**Profit-seeking ("rational") behavior**~~

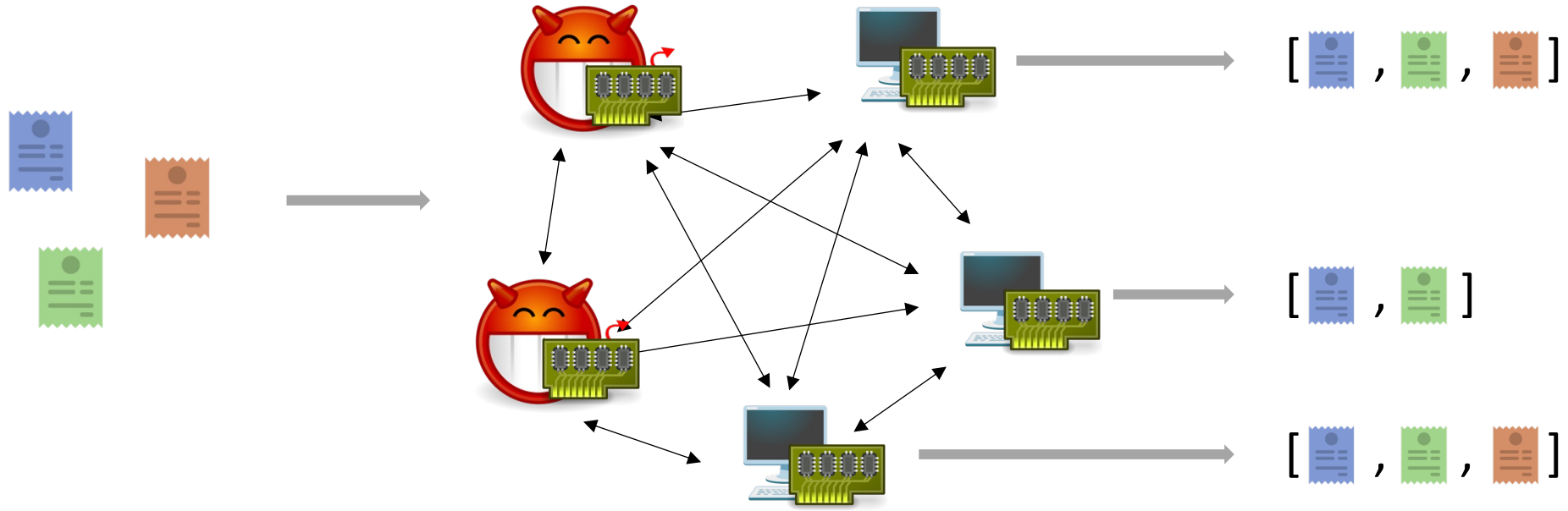
  ~~Nodes seek to maximize their payouts from protocol rewards etc.~~

Incentives in consensus are a can of worms …

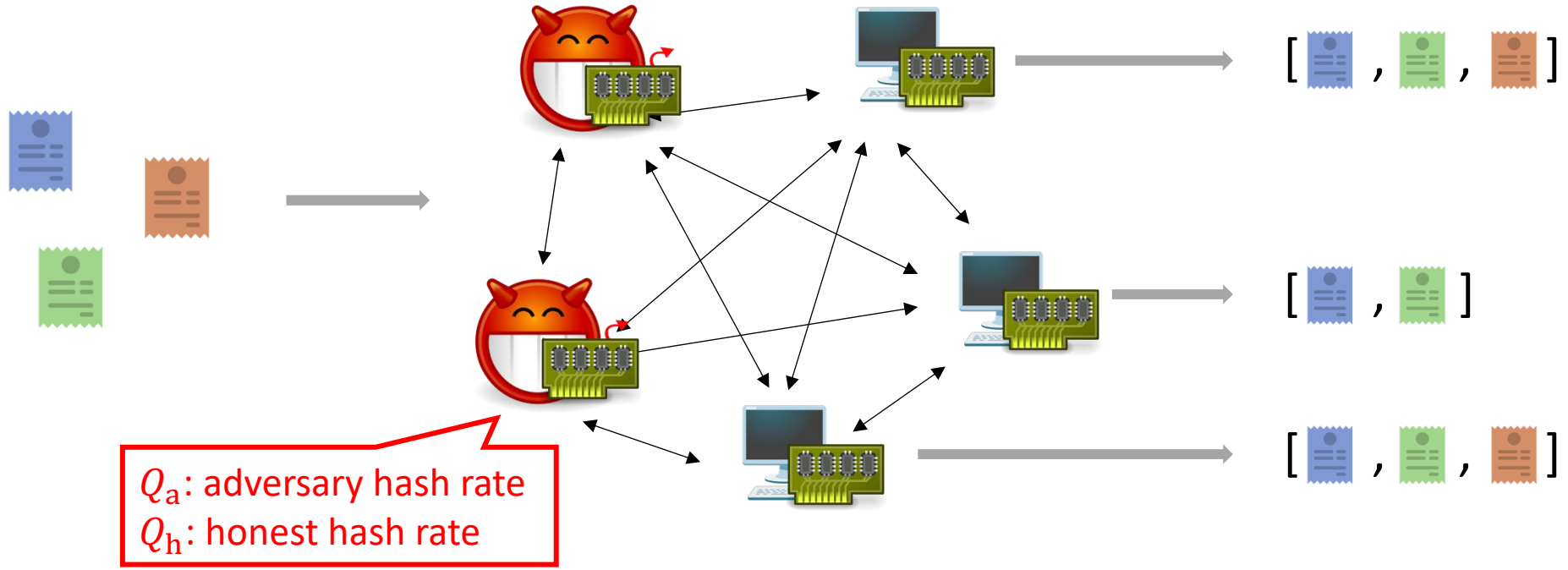
SMR Example 1: Proof-of-Work Nakamoto Consensus



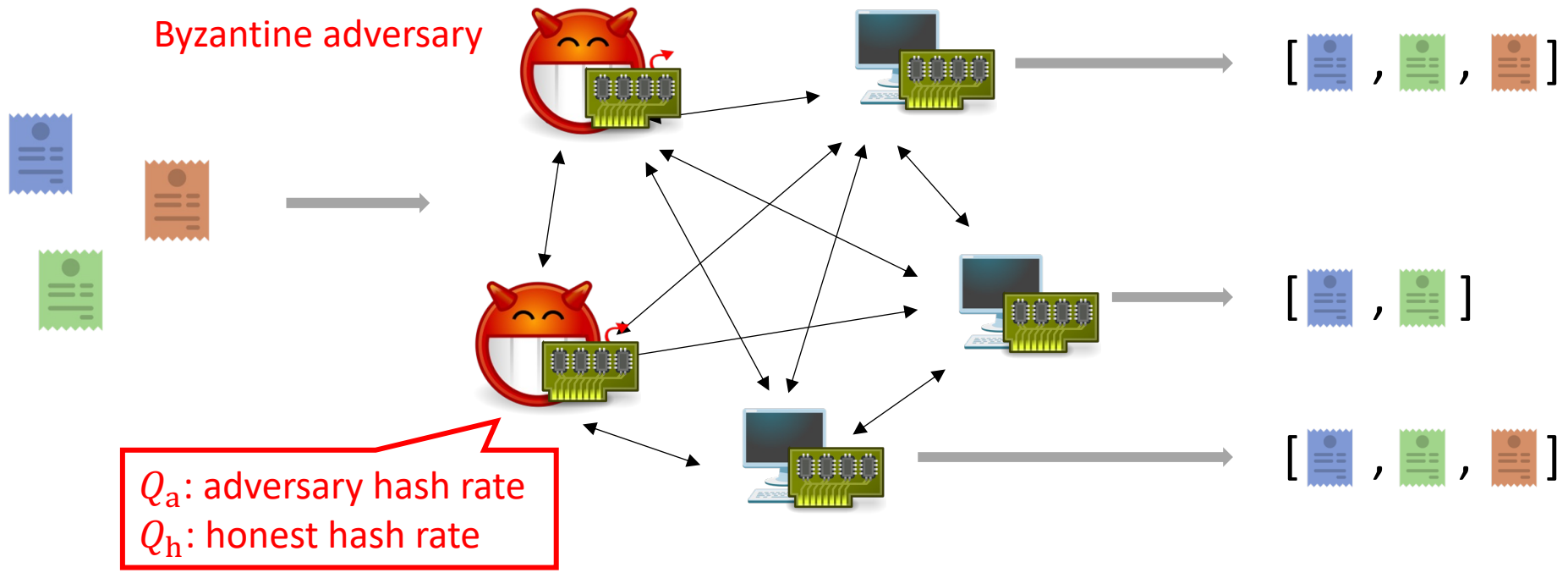
SMR Example 1: Proof-of-Work Nakamoto Consensus



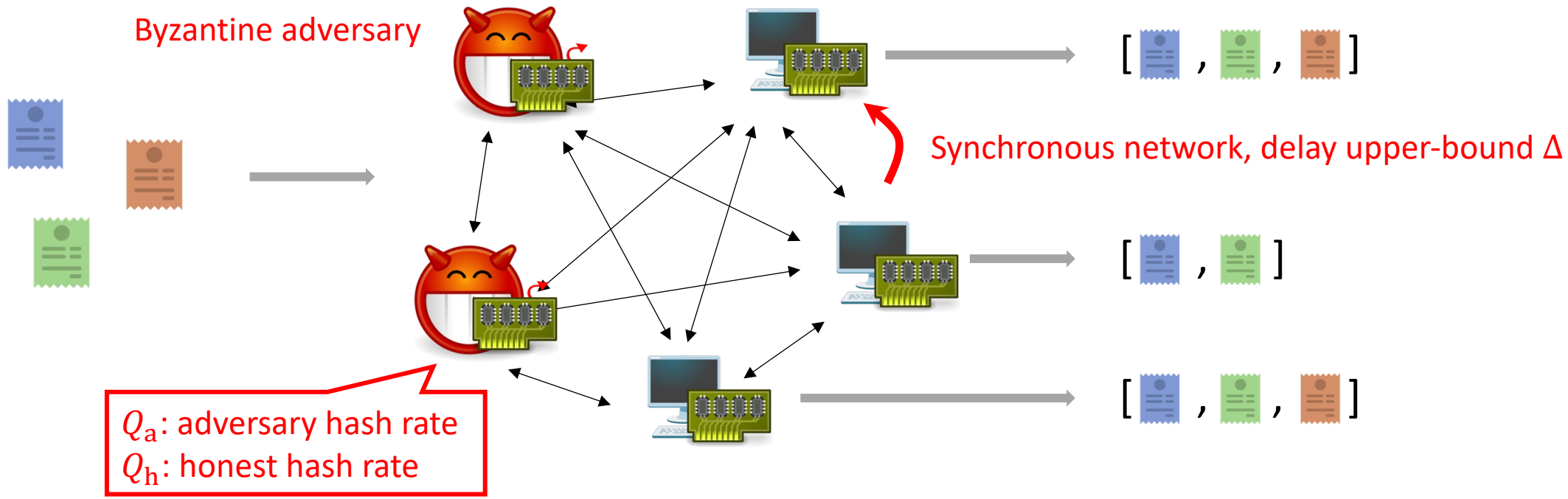
SMR Example 1: Proof-of-Work Nakamoto Consensus



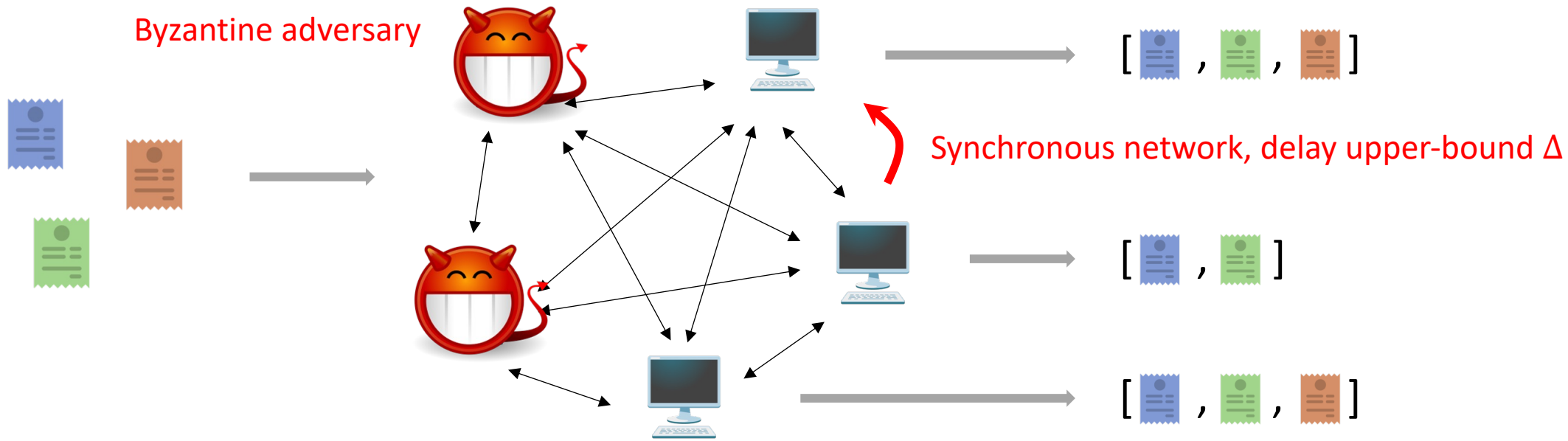
SMR Example 1: Proof-of-Work Nakamoto Consensus



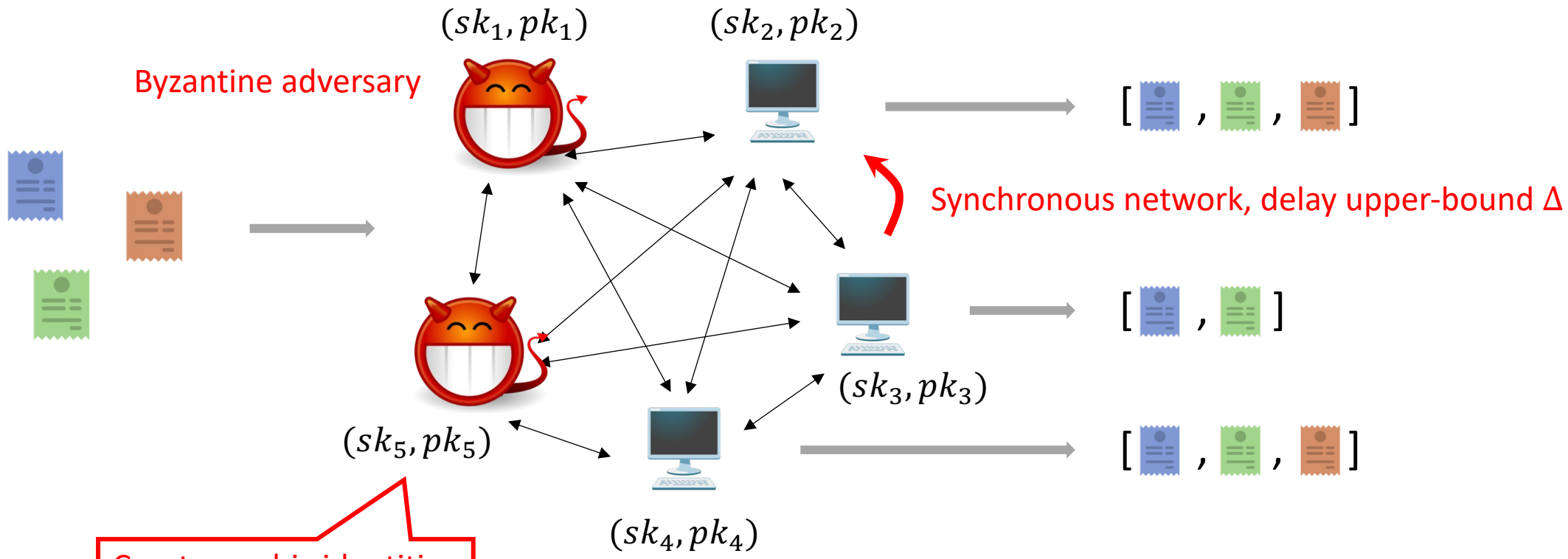
SMR Example 1: Proof-of-Work Nakamoto Consensus



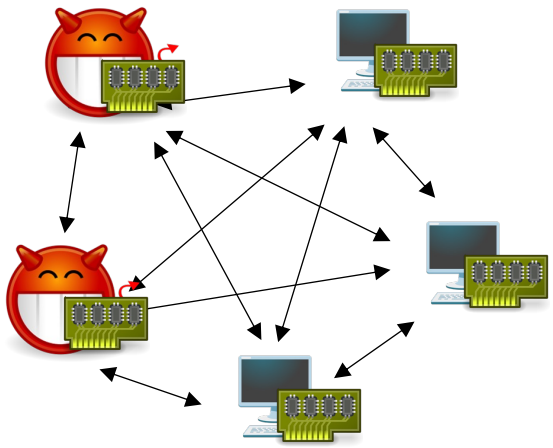
SMR Example 1: Permissioned Nakamoto Consensus



SMR Example 1: Permissioned Nakamoto Consensus



Nakamoto Consensus



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

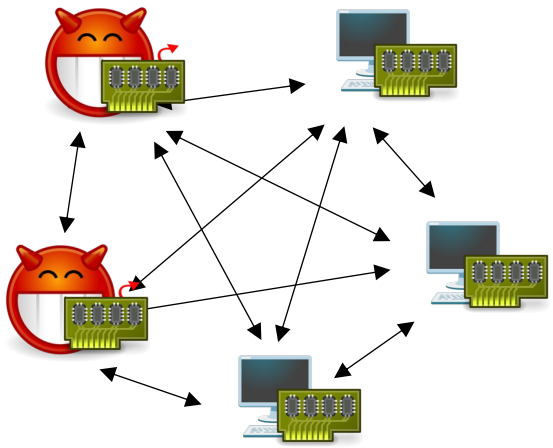
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

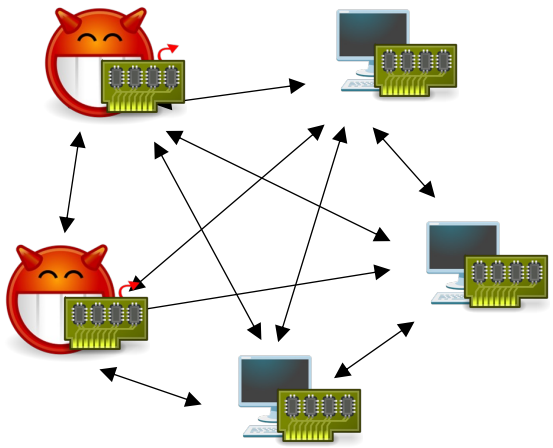
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Genesis block

Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

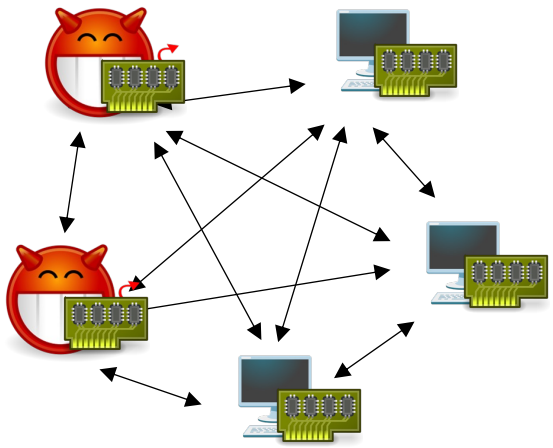
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Genesis block

Nakamoto Consensus

Genesis block

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

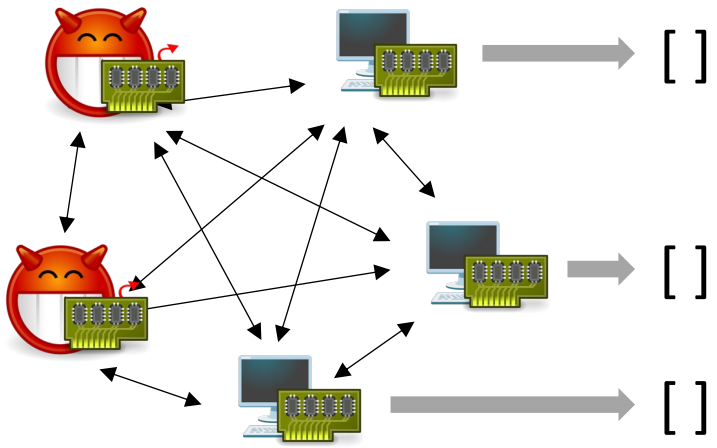
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

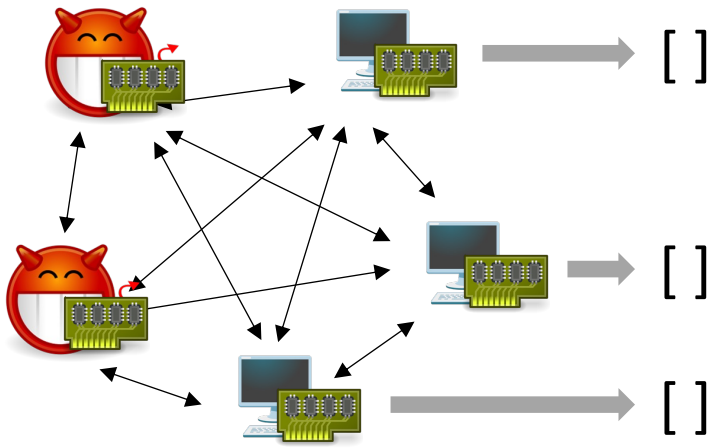
$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

"Mining"



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

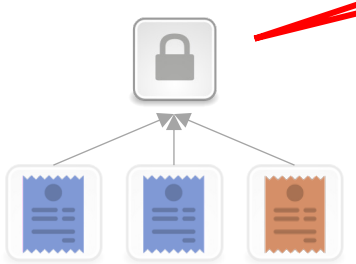
$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

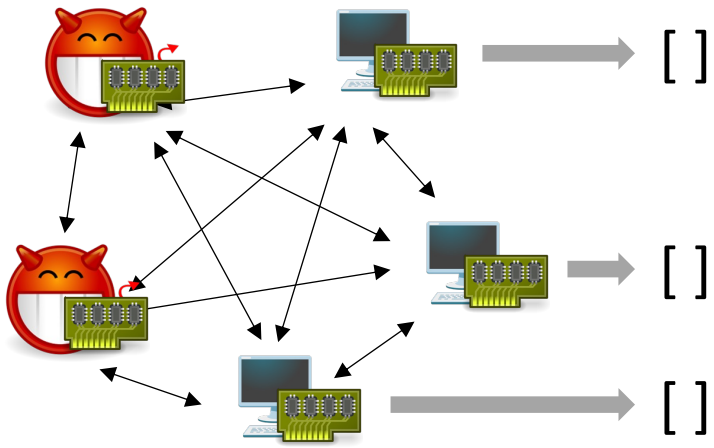
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

"Mining"



Genesis block



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

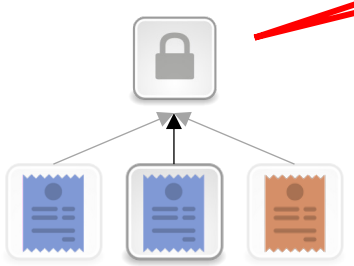
$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

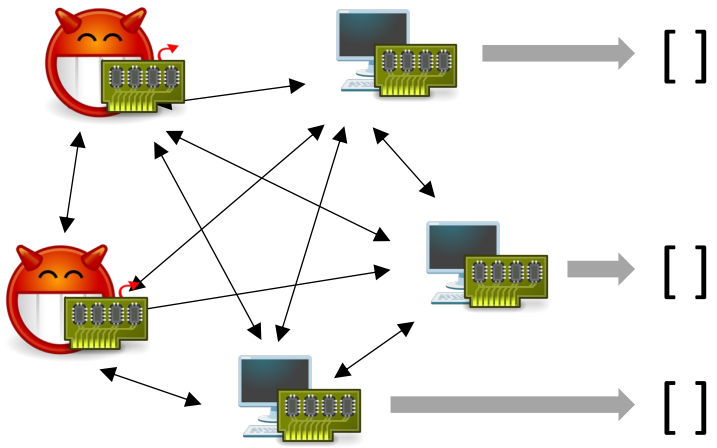
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

"Mining"



Genesis block



$Hash: \{0,1\}^* \rightarrow [1, 2^{256}]$

Output of $Hash$ assumed to be random

→ $\lambda_{\text{a}} = Q_{\text{a}} \cdot D$: adversary block production rate

→ $\lambda_{\text{h}} = Q_{\text{h}} \cdot D$: honest block production rate

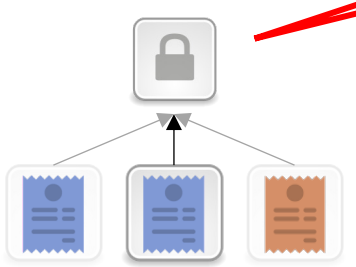
Q_{a}: adversary hash rate

Q_{h}: honest hash rate

Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

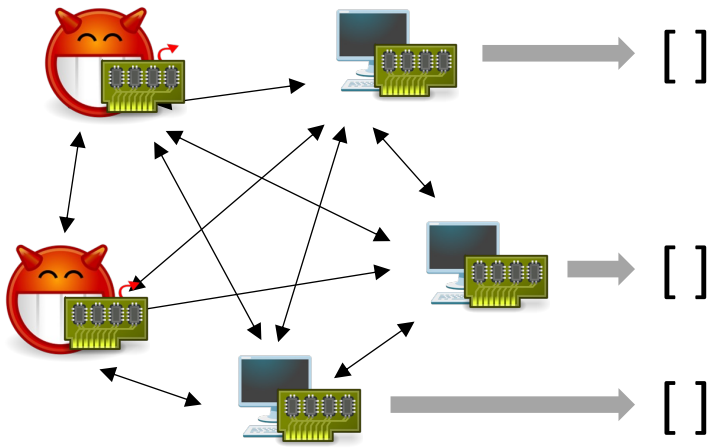
$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

"Mining"



$Hash: \{0,1\}^* \rightarrow [1, 2^{256}]$

Output of $Hash$ assumed to be random

→ $\lambda_{\text{a}} = Q_{\text{a}} \cdot D$: adversary block production rate

→ $\lambda_{\text{h}} = Q_{\text{h}} \cdot D$: honest block production rate

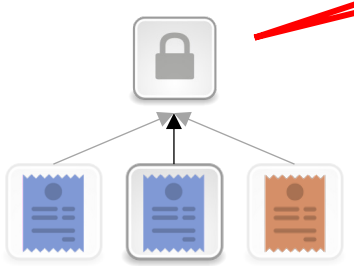
Q_{a}: adversary hash rate

Q_{h}: honest hash rate

Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

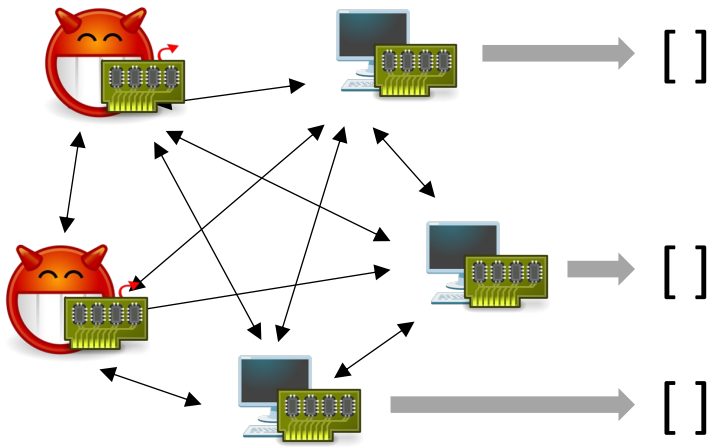
$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

"Mining"



$Hash: \{0,1\}^* \rightarrow [1, 2^{256}]$

Output of $Hash$ assumed to be random

→ $\lambda_{\text{a}} = Q_{\text{a}} \cdot D$: adversary block production rate

→ $\lambda_{\text{h}} = Q_{\text{h}} \cdot D$: honest block production rate

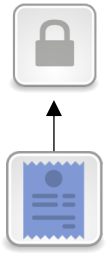
Q_{a}: adversary hash rate

Q_{h}: honest hash rate

Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

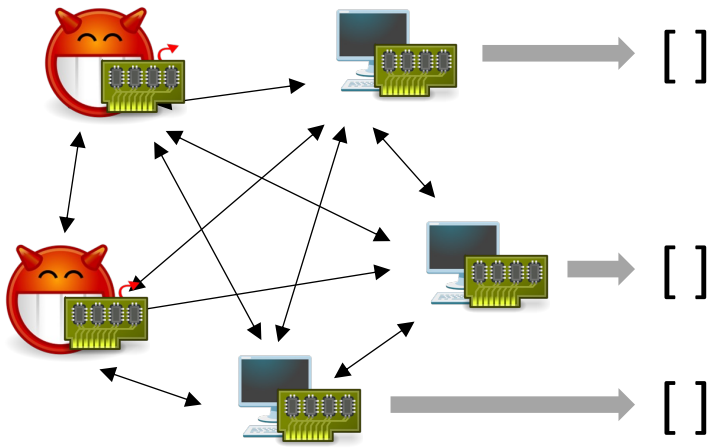
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

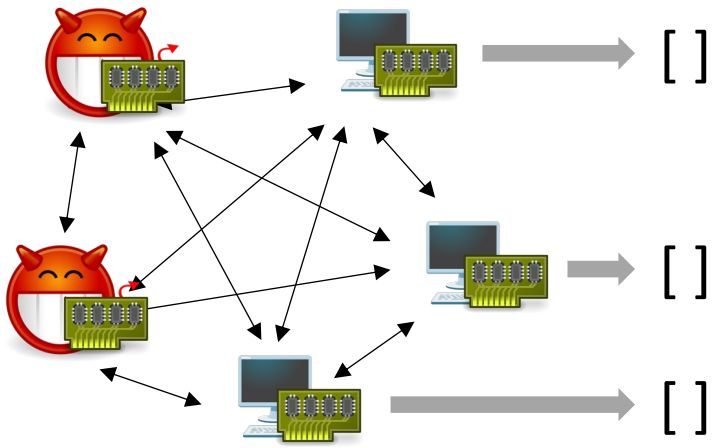
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

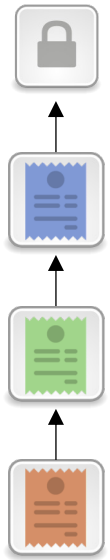
broadcast b_{new}



Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

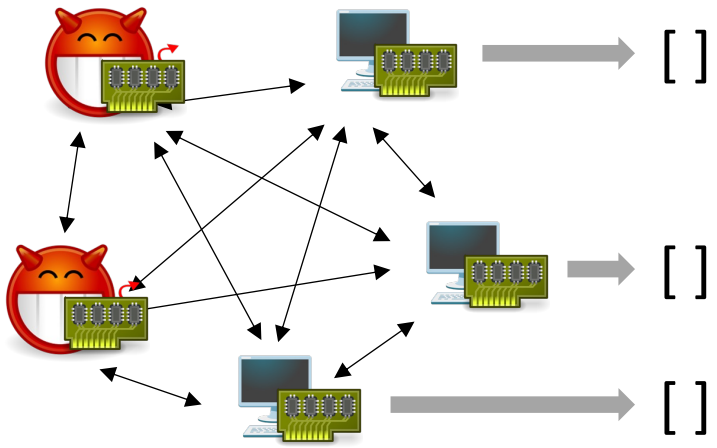
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

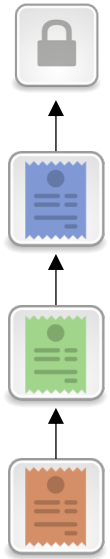
broadcast b_{new}



Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

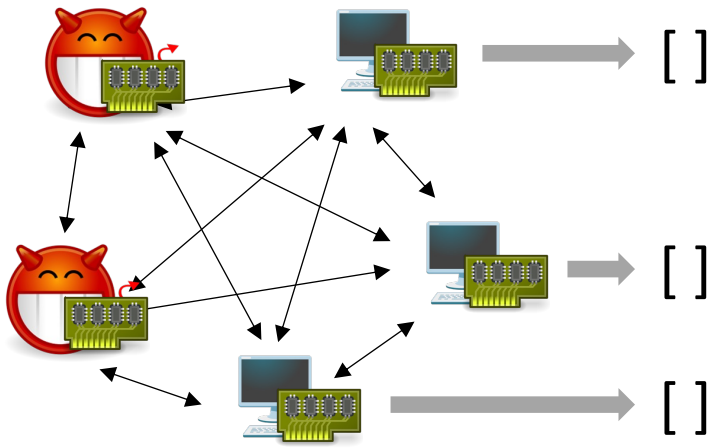
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

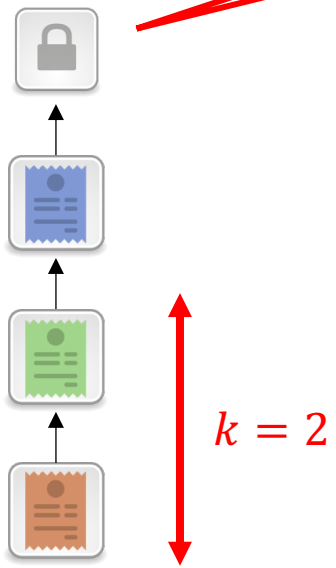
broadcast b_{new}



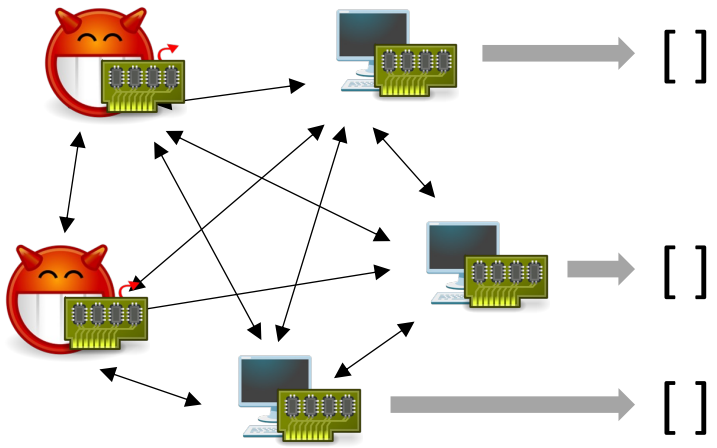
Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



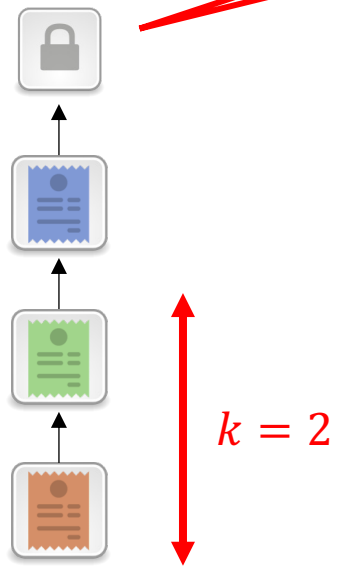
Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



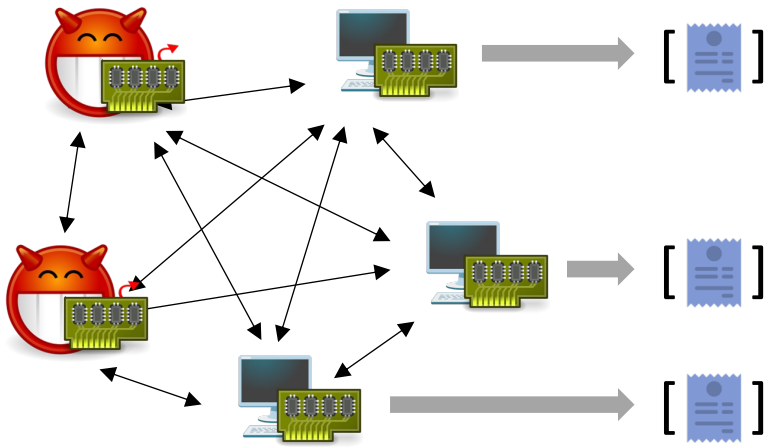
Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



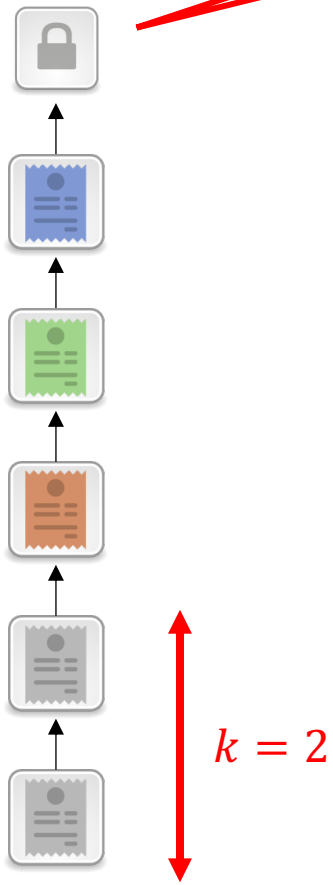
Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Nakamoto Consensus

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

Genesis block



Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

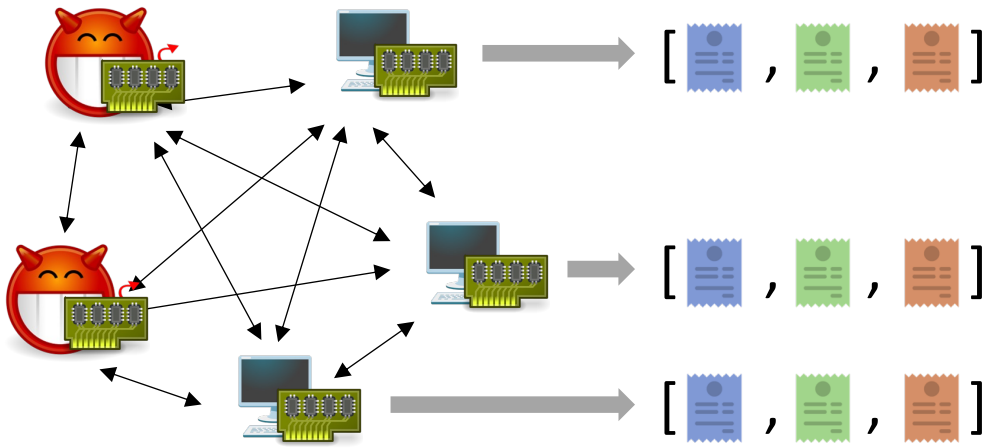
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

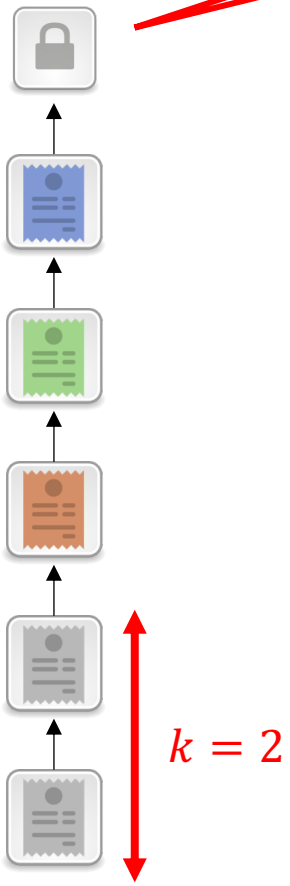
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

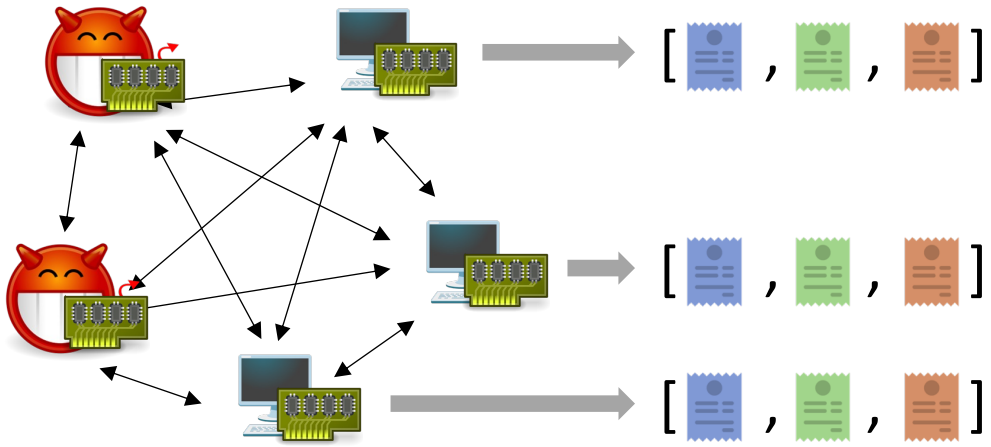
broadcast b_{new}

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_{\text{a}} > Q_{\text{h}}$, then $\lambda_{\text{a}} > \lambda_{\text{h}}$ …

Genesis block





Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

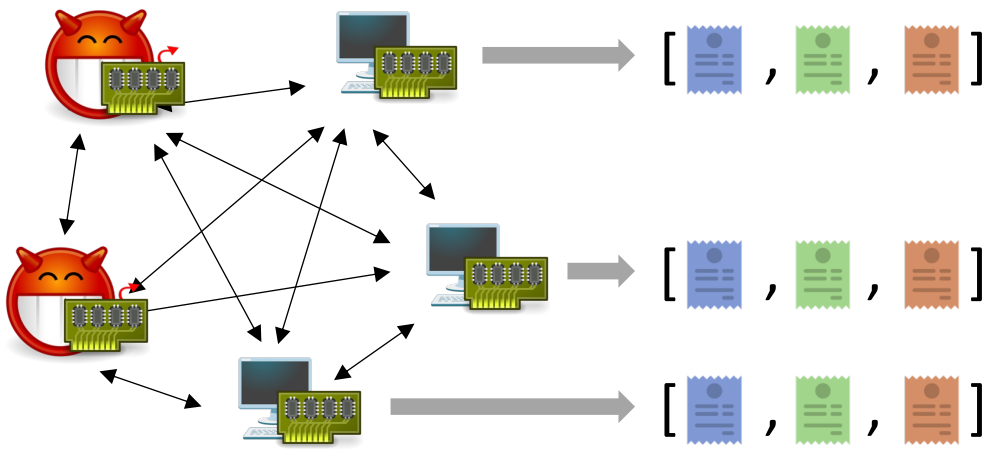
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

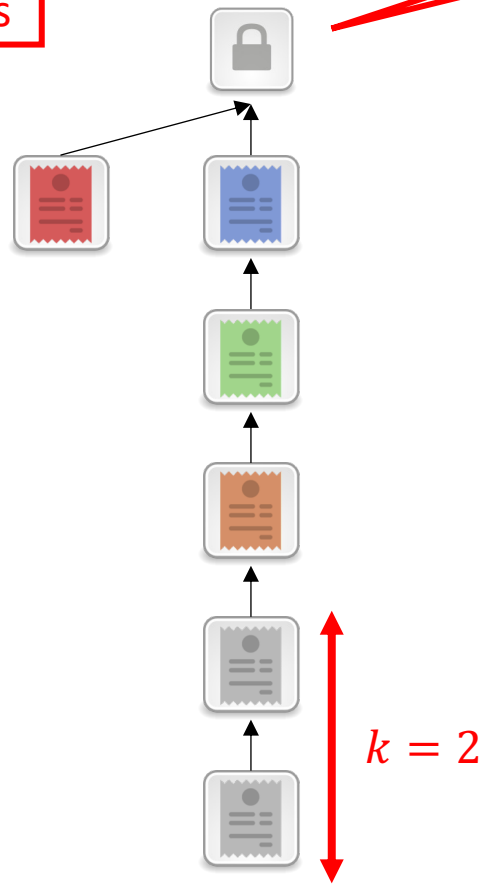
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



$k = 2$

Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

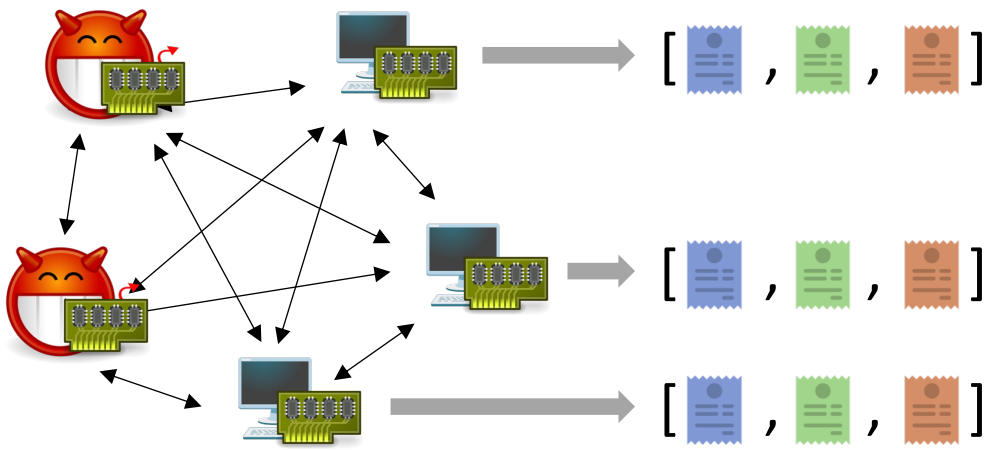
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

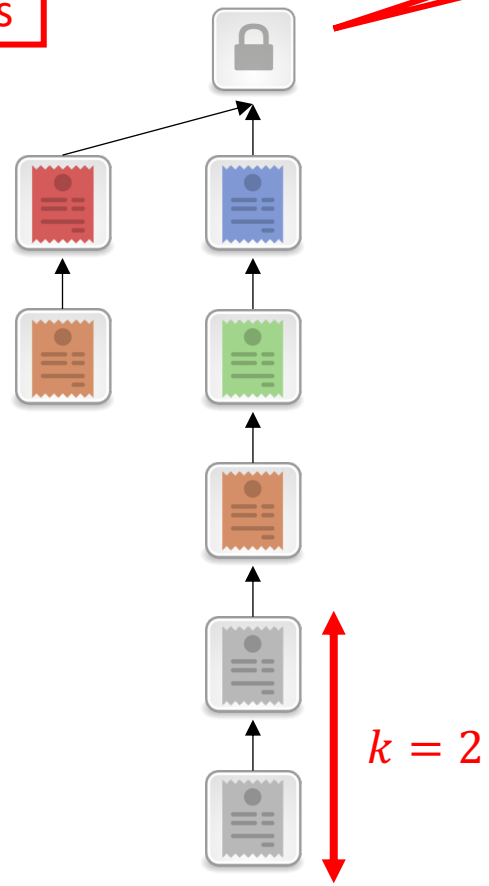
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



$k = 2$

Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

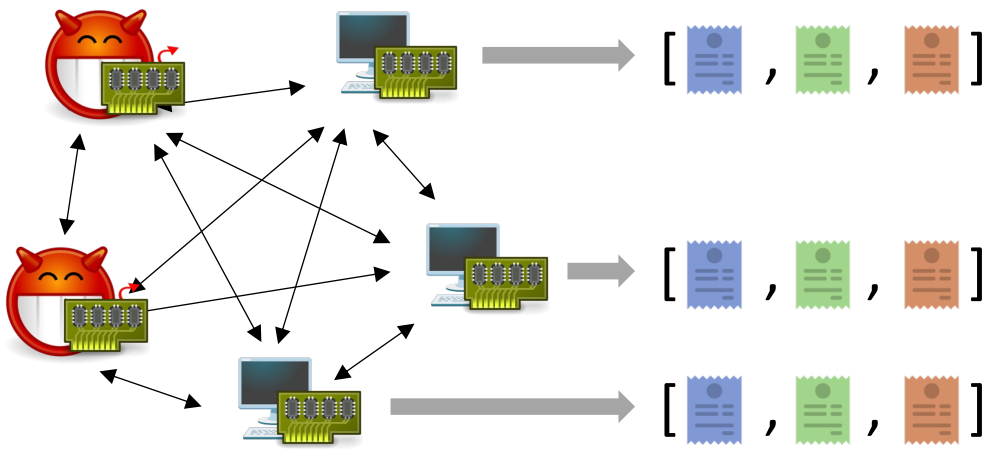
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

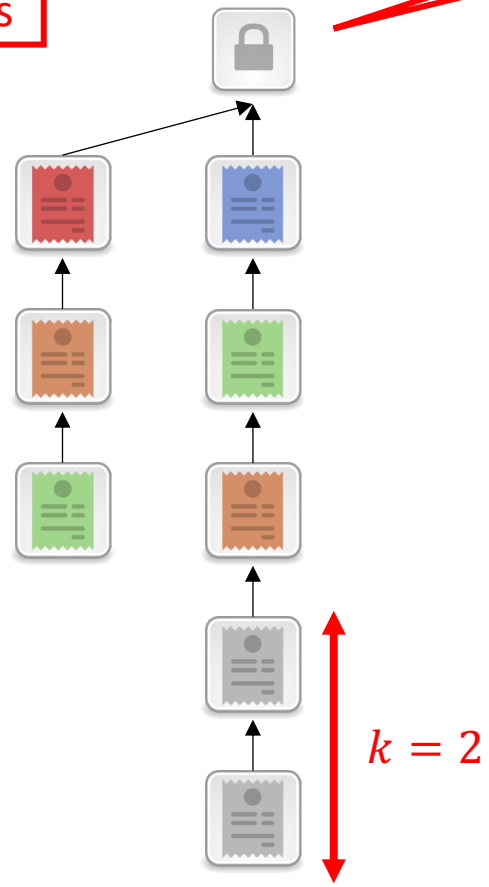
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



$k = 2$

Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

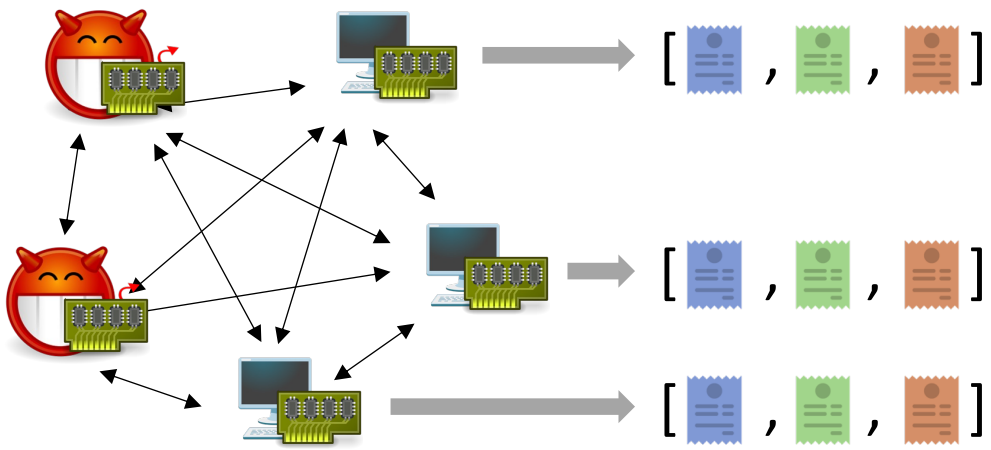
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

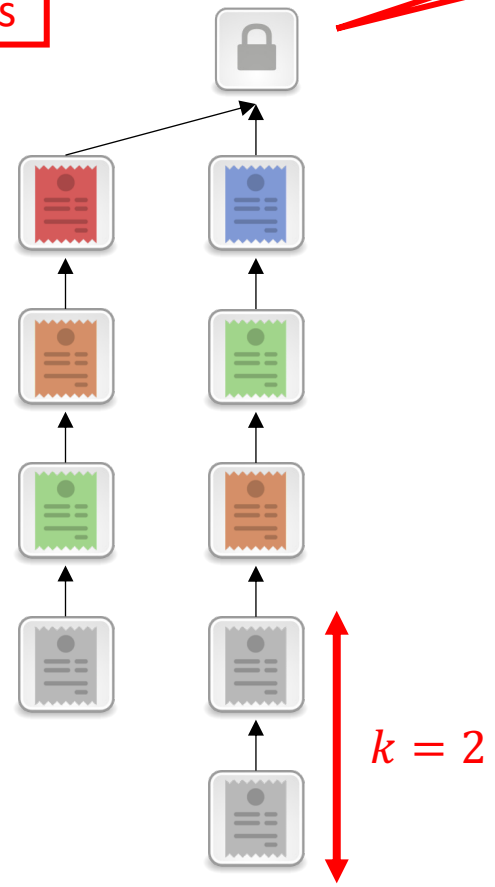
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



$k = 2$

Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

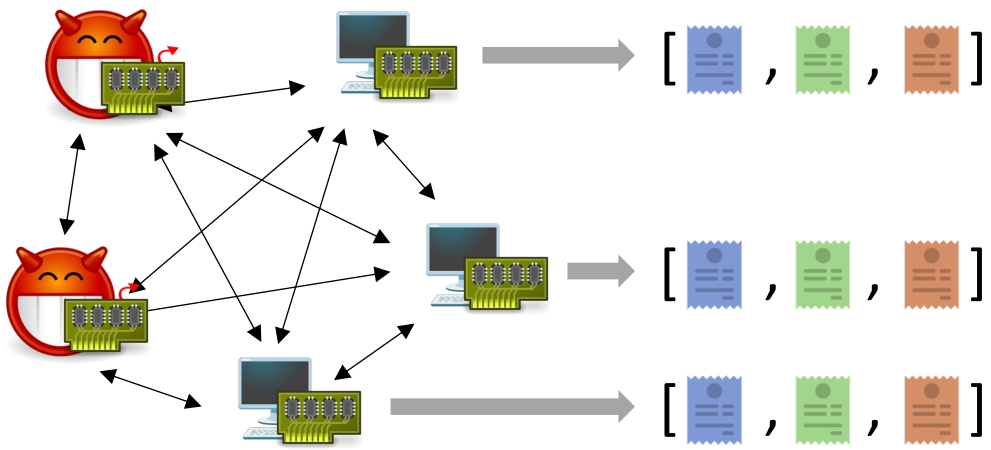
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

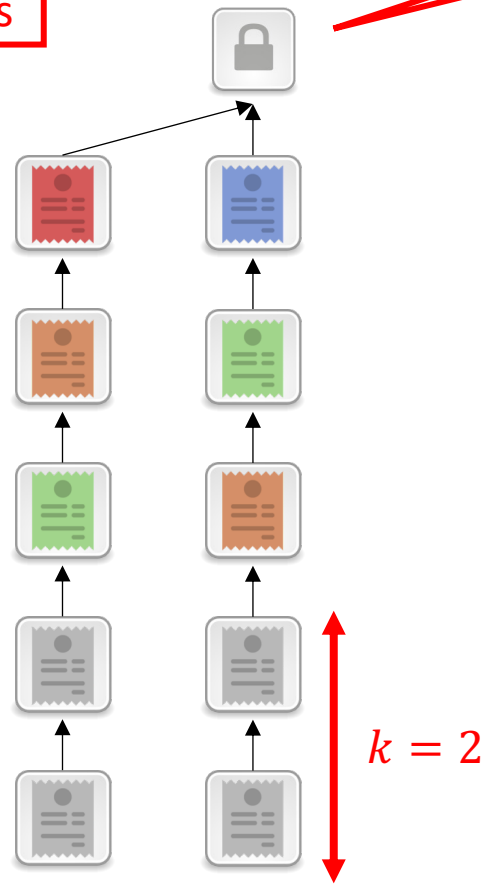
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

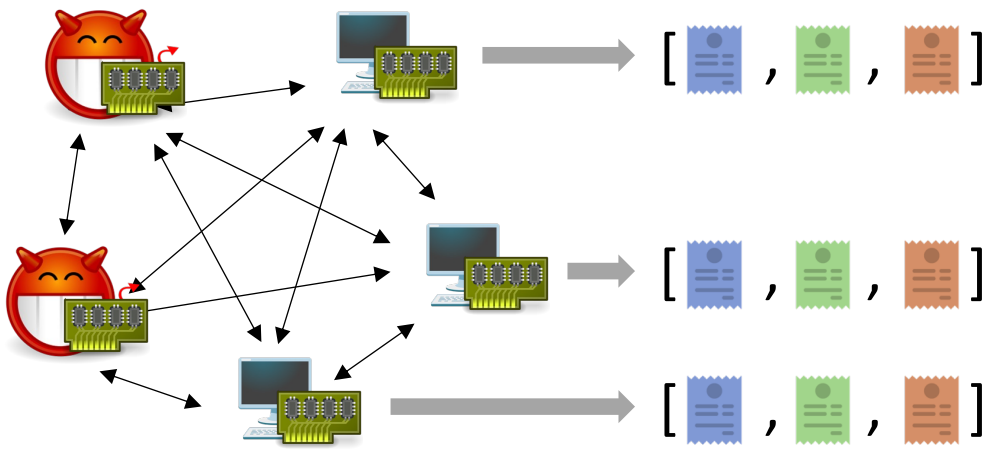
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

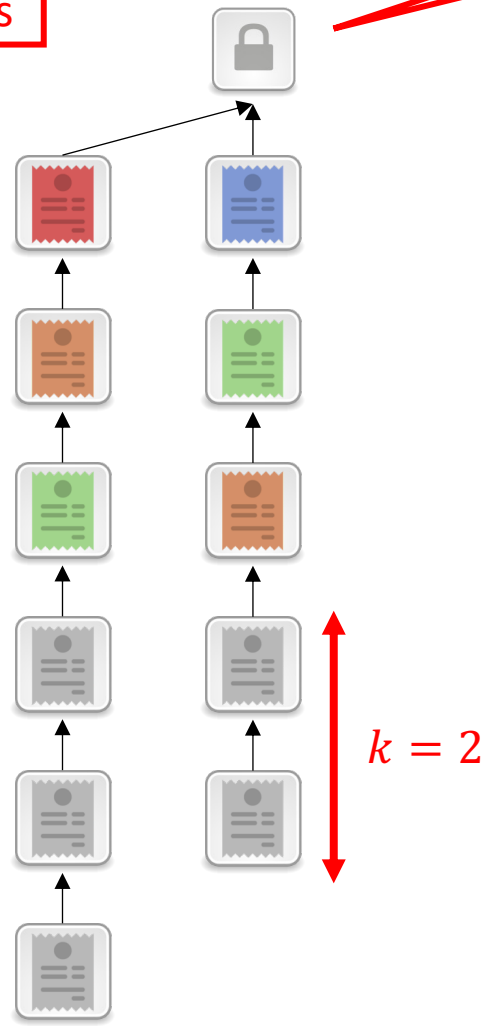
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

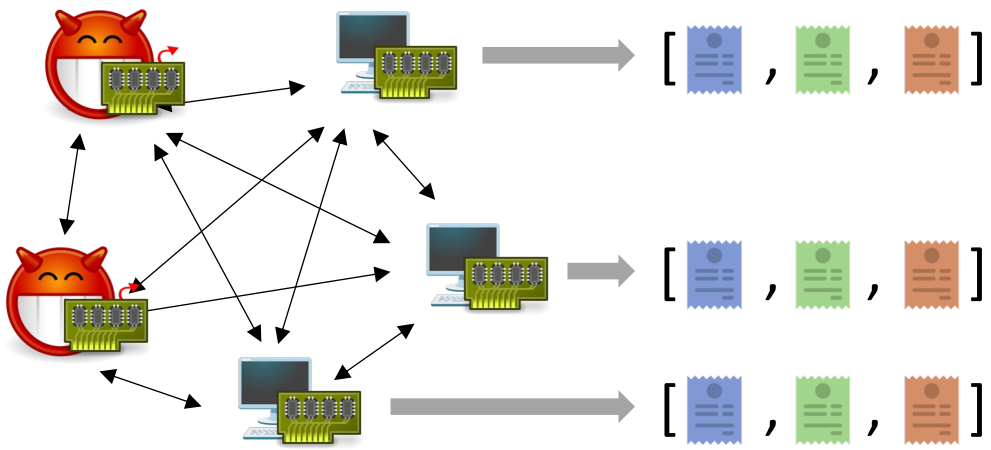
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

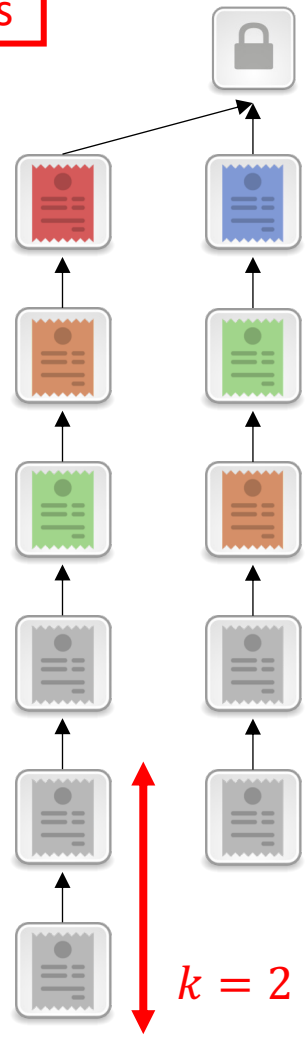
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

Genesis block



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

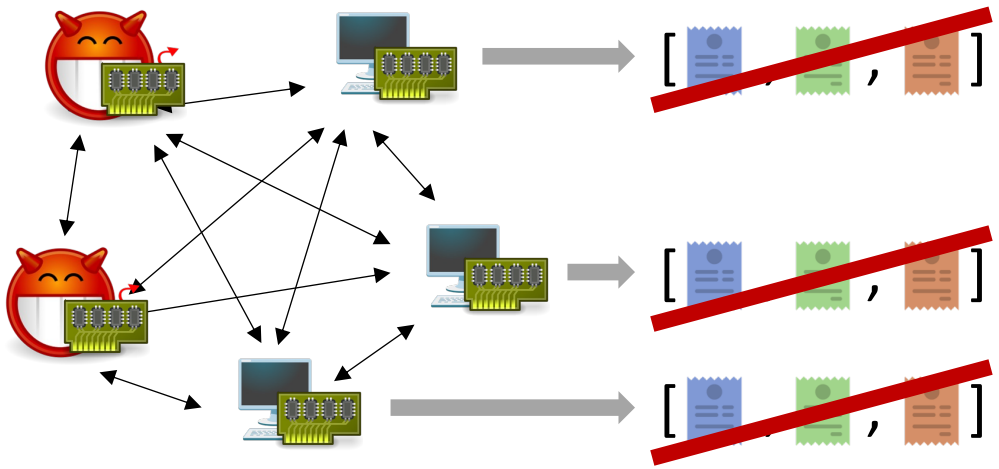
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

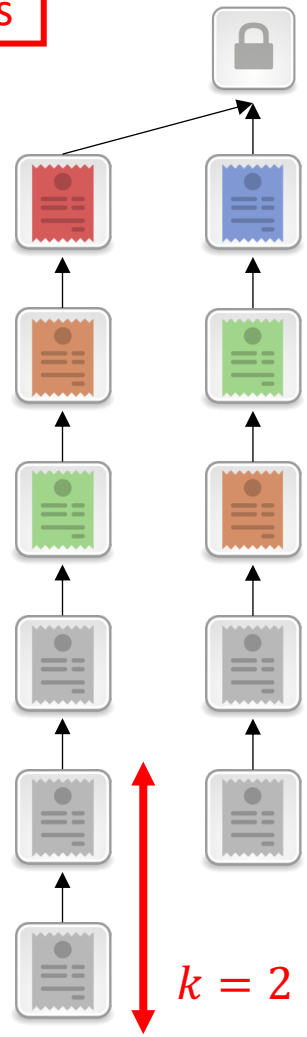
broadcast b_{new}



Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$,
then $\lambda_a > \lambda_h$ …

Genesis block



Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

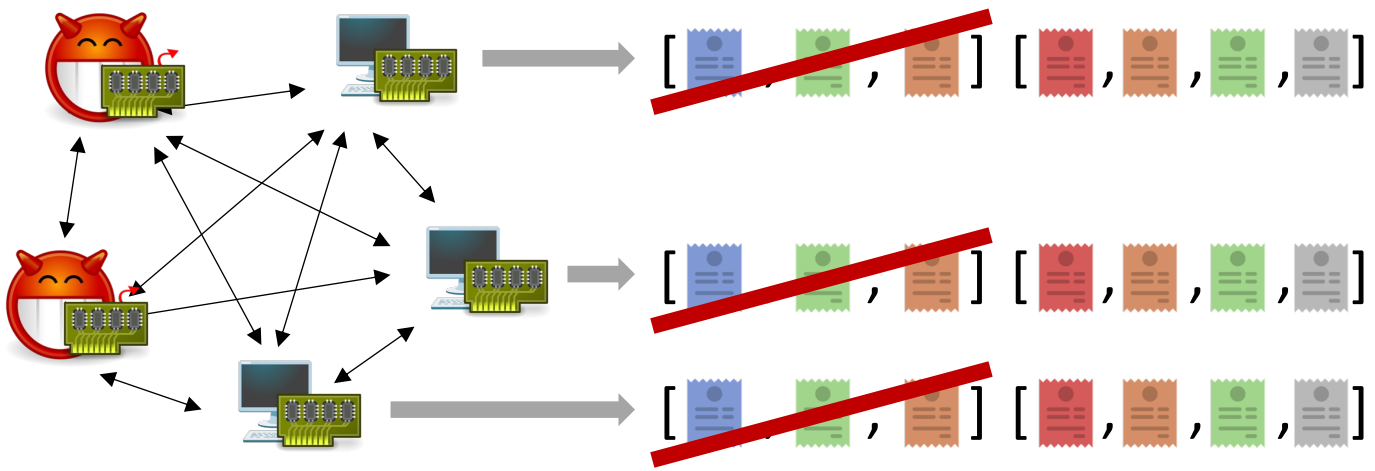
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

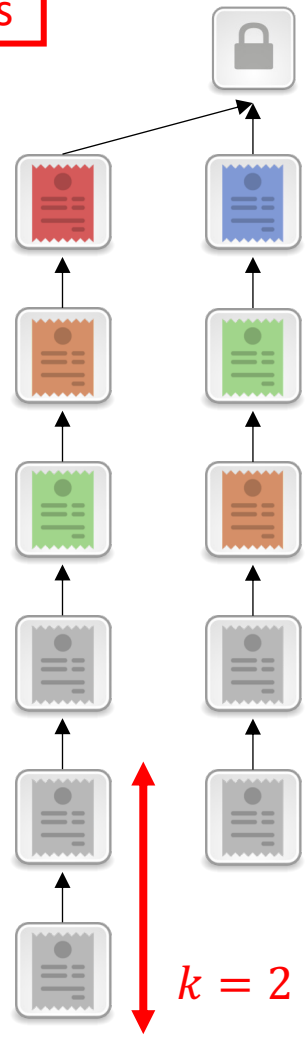
broadcast b_{new}

Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$,
then $\lambda_a > \lambda_h$ …

Genesis block





Nakamoto Consensus

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

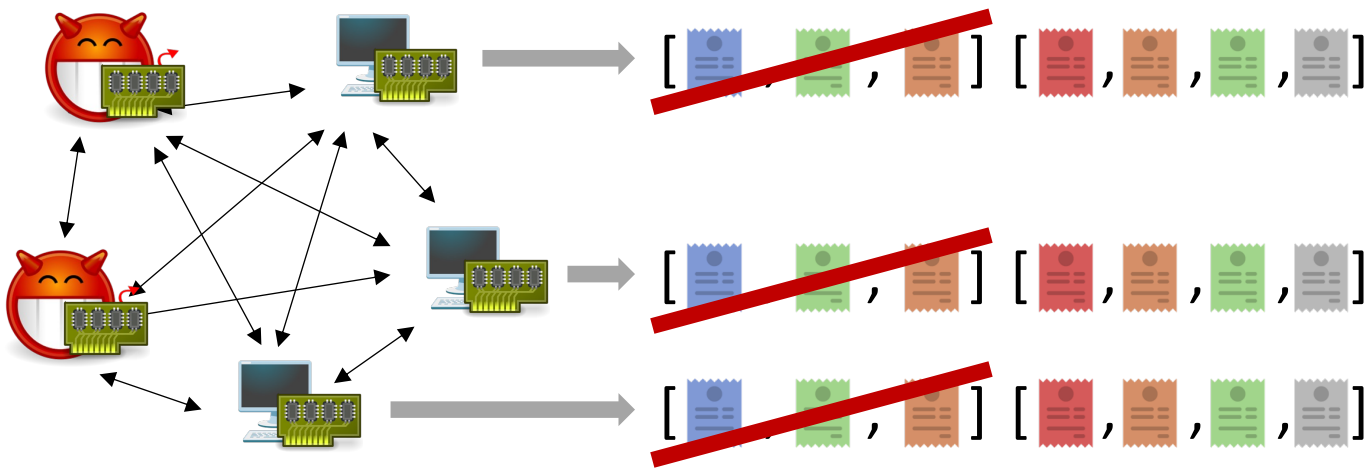
broadcast b_{new}

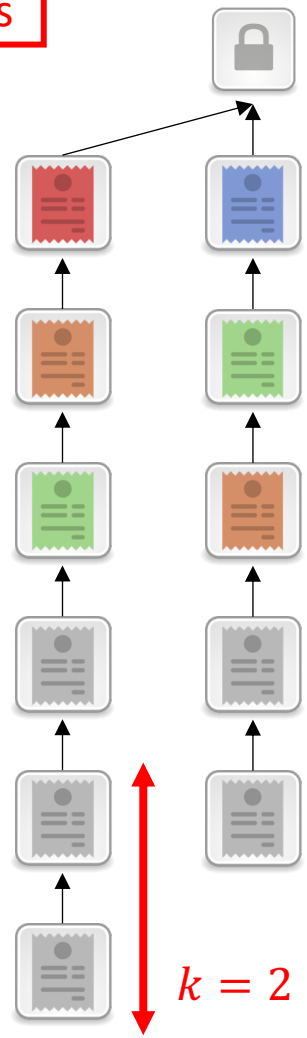
Assume $\Delta = 0$.
→ Honest nodes see the same blocks

If $Q_a > Q_h$, then $\lambda_a > \lambda_h$ …

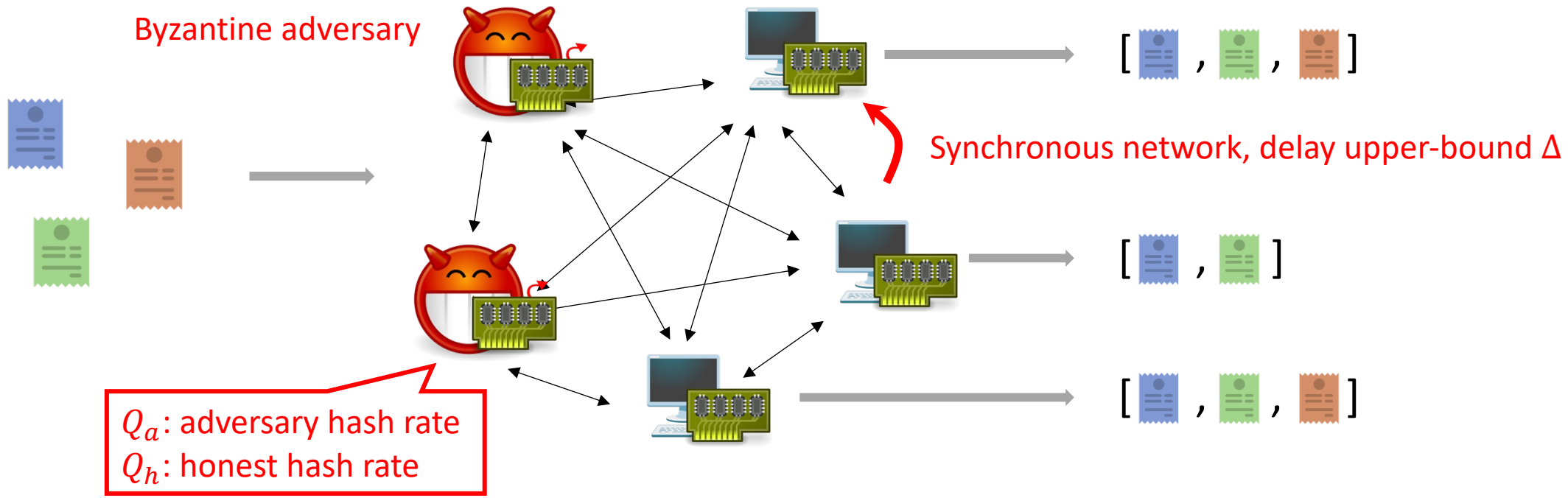
"Private attack"

Genesis block





SMR Example 1: Proof-of-Work Nakamoto Consensus

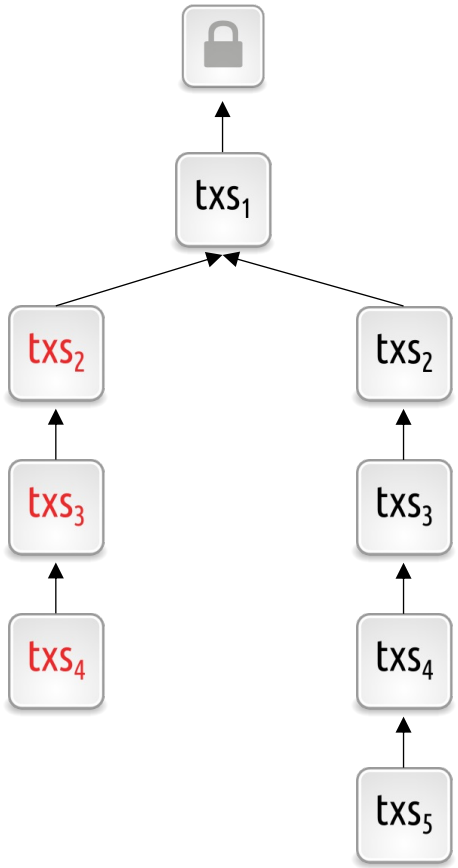


Theorem

For $\Delta = 0$, proof-of-work Nakamoto consensus is safe and live iff $Q_{\mathrm{a}} < Q_{\mathrm{h}}$, under synchrony, except with probability decaying exponentially in k.

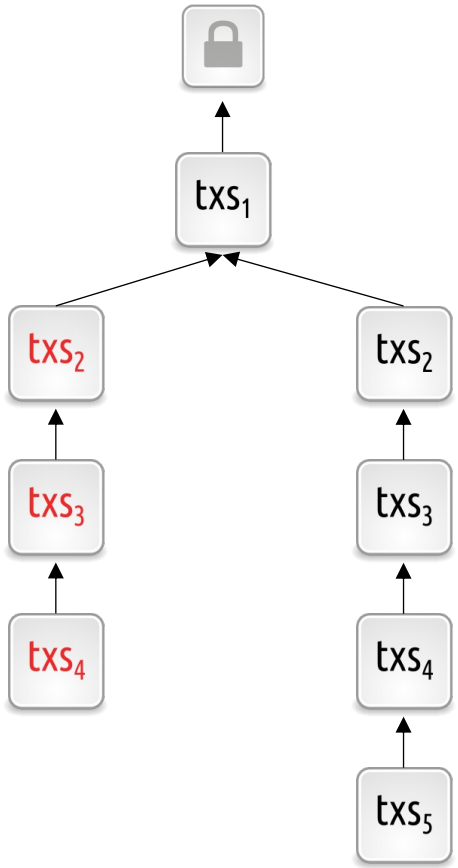
"The Sleepy Model of Consensus", Pass, Shi 2017; "Everything is a Race and Nakamoto Always Wins", Dembo et al. 2020

Security for $\Delta = 0$



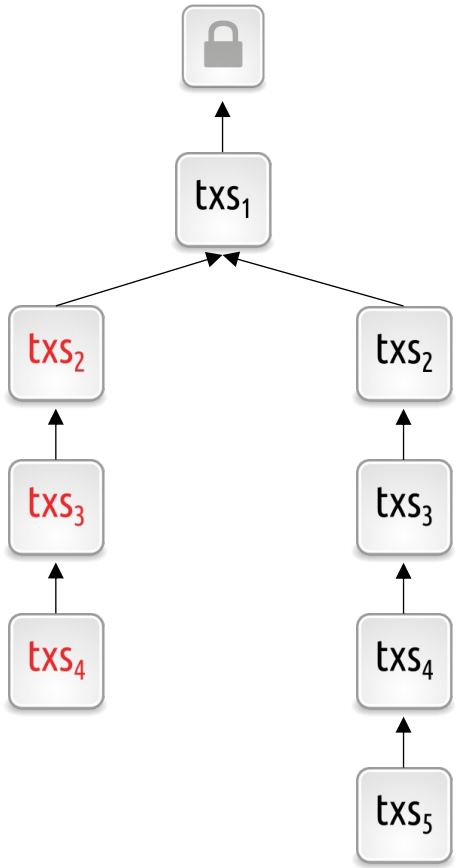
"The Sleepy Model of Consensus", Pass, Shi 2017; "Everything is a Race and Nakamoto Always Wins", Dembo et al. 2020

Security for $\Delta = 0$



$$\lambda_{\text{growth}} = \lambda_{\text{h}} = Q_{\text{h}} \cdot D$$

Security for $\Delta = 0$

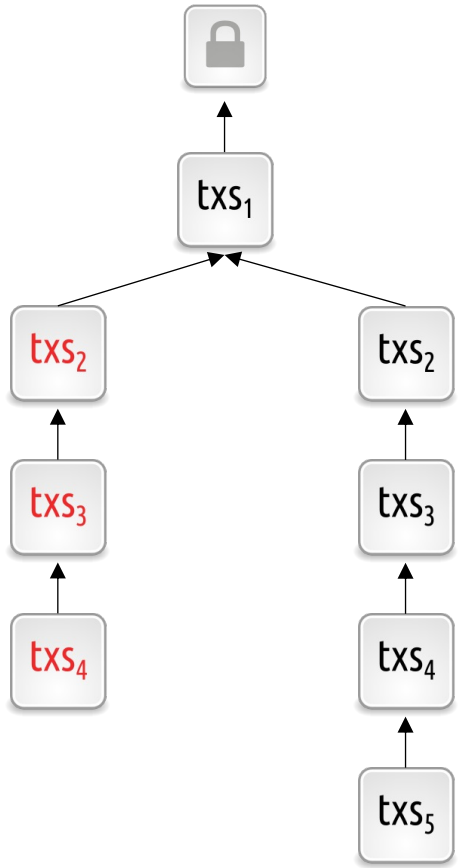


$\lambda_{\mathrm{a}} = Q_{\mathrm{a}} \cdot D$

$\lambda_{\mathrm{growth}} = \lambda_{\mathrm{h}} = Q_{\mathrm{h}} \cdot D$

Security for $\Delta = 0$

Under $\Delta = 0, Q_{\mathrm{a}} < Q_{\mathrm{h}}$:



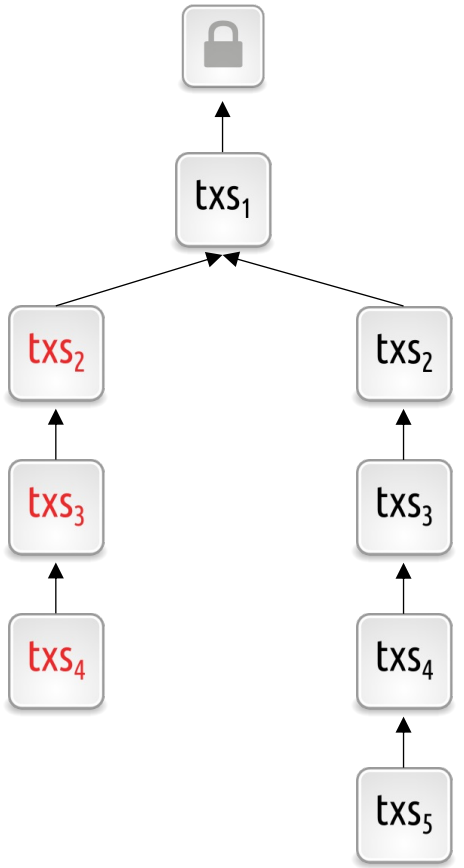
$$\lambda_{\mathrm{a}} = Q_{\mathrm{a}} \cdot D \quad < \quad \lambda_{\mathrm{growth}} = \lambda_{\mathrm{h}} = Q_{\mathrm{h}} \cdot D$$

Security for $\Delta = 0$

Under $\Delta = 0, Q_{\mathrm{a}} < Q_{\mathrm{h}}$:

Safety:

- Longest chain grows faster than adversary chain.
- Safety violation requires that adversary chain catches up with longest chain at length k, or length $k + 1$, or length $k + 2$, etc.
- → Probability decays exponentially in k.



$$\lambda_{\mathrm{a}} = Q_{\mathrm{a}} \cdot D \quad < \quad \lambda_{\mathrm{growth}} = \lambda_{\mathrm{h}} = Q_{\mathrm{h}} \cdot D$$

Security for $\Delta = 0$

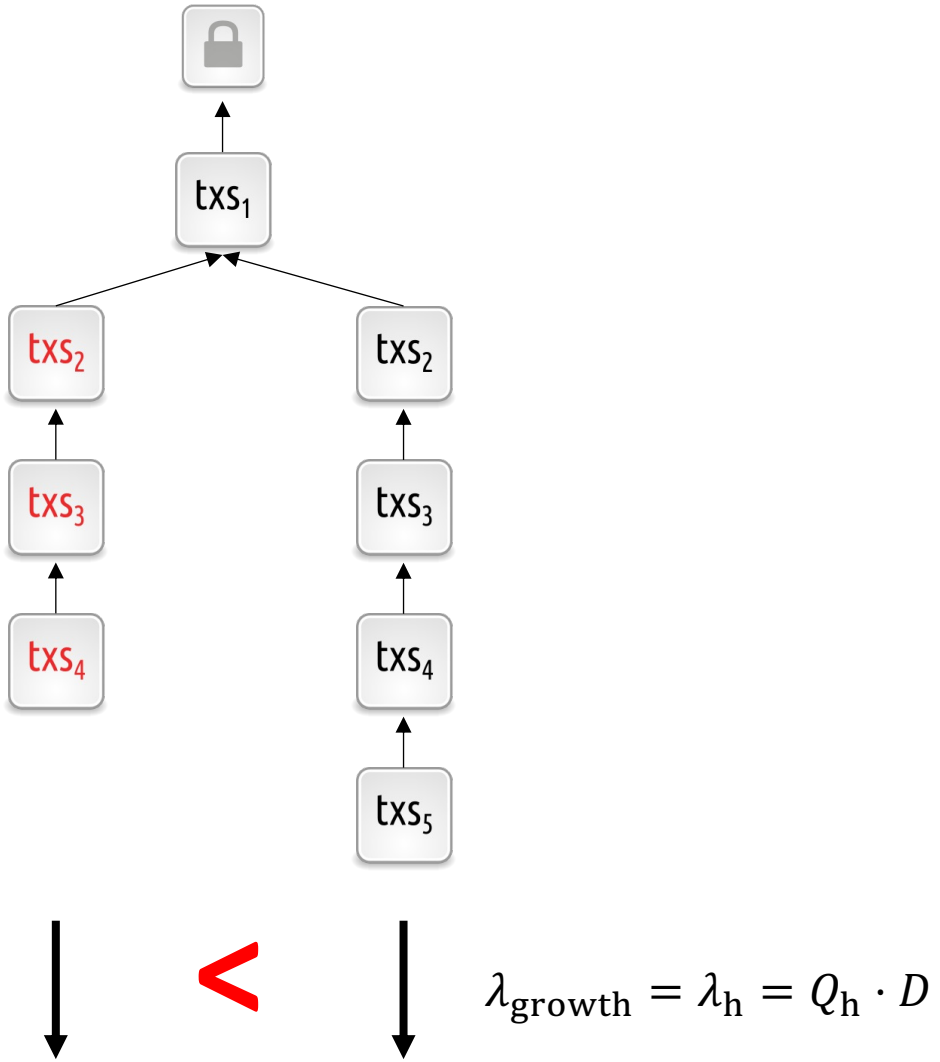
Under $\Delta = 0, Q_{\mathrm{a}} < Q_{\mathrm{h}}$:

Safety:

- Longest chain grows faster than adversary chain.
- Safety violation requires that adversary chain catches up with longest chain at length k, or length $k + 1$, or length $k + 2$, etc.
- → Probability decays exponentially in k.

Liveness:

- Longest chain grows at rate $\lambda_{\mathrm{growth}} = \lambda_{\mathrm{h}} > \lambda_{\mathrm{a}}$.
- By pigeonhole principle, there must be honestly produced blocks on the longest chain.
- → Transactions get confirmed eventually.



Security for $\Delta > 0$

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

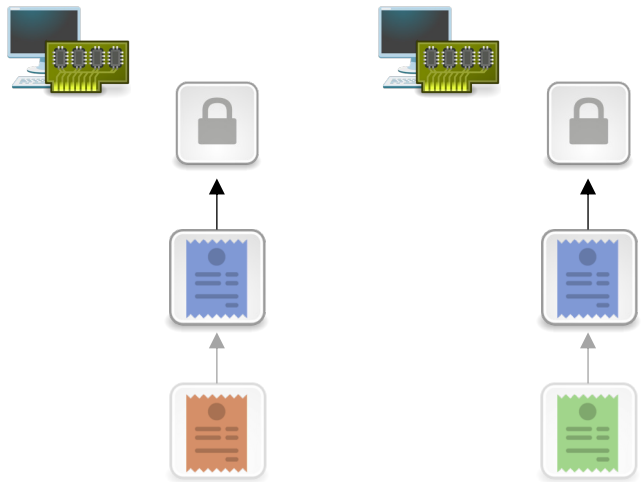
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Time

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

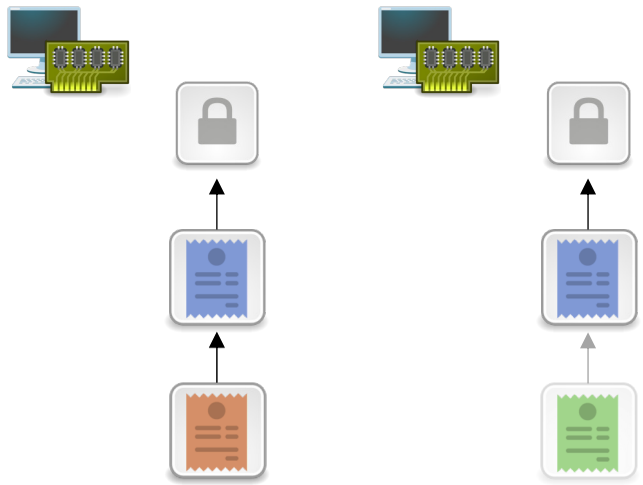
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}



Security for $\Delta > 0$

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

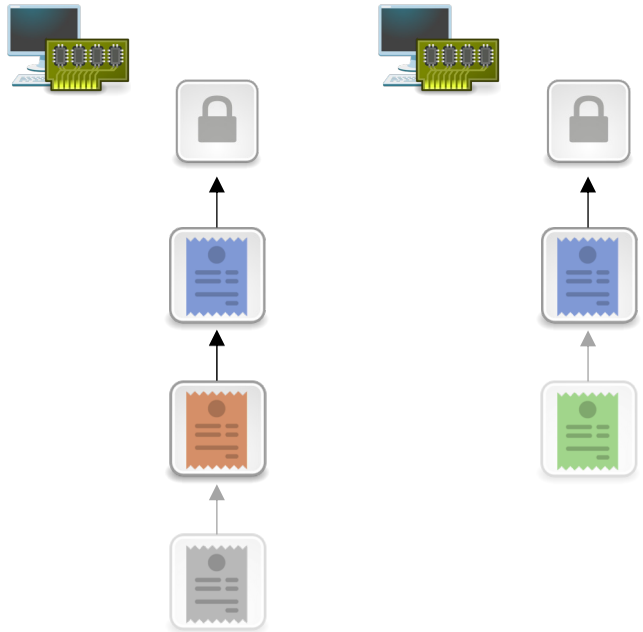
until new block arrives:

$\pi \leftarrow RandomNonce()$

$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

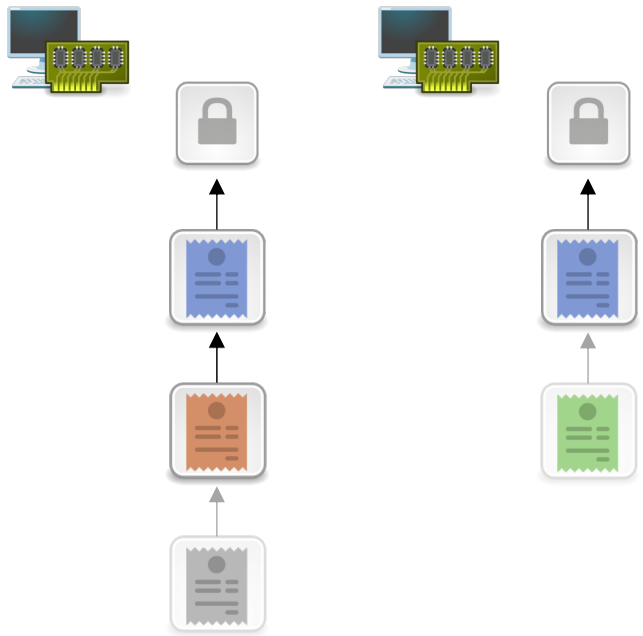
broadcast b_{new}

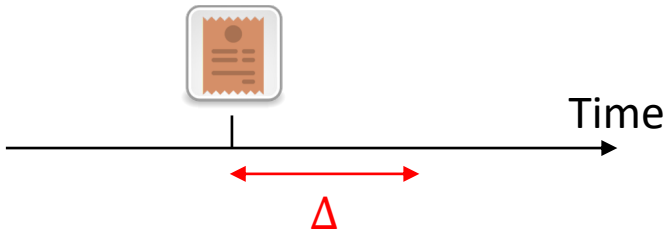


Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
$b^* \leftarrow GetLongestChain()$
$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
$txs \leftarrow GetPendingTxs()$
until new block arrives:
$\pi \leftarrow RandomNonce()$
$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
broadcast b_{new}

Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!

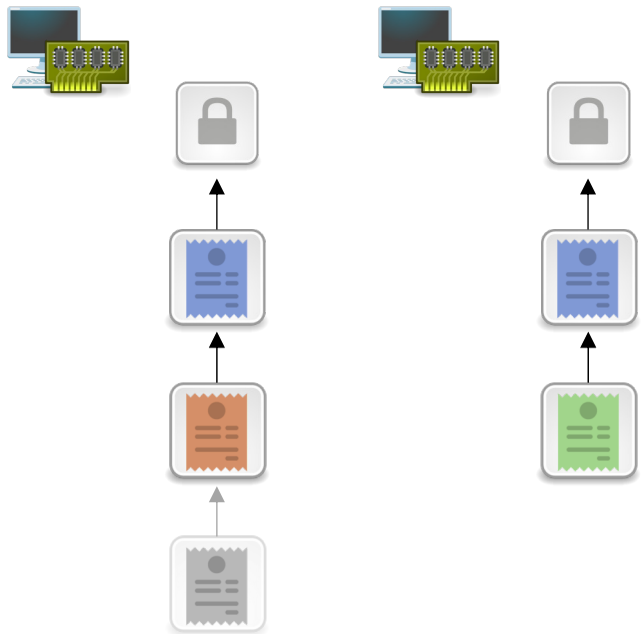


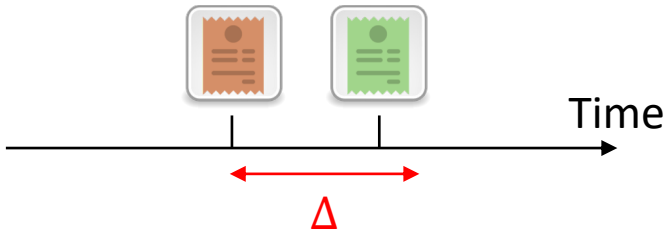


Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
$b^* \leftarrow GetLongestChain()$
$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
$txs \leftarrow GetPendingTxs()$
until new block arrives:
$\pi \leftarrow RandomNonce()$
$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
broadcast b_{new}

Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!

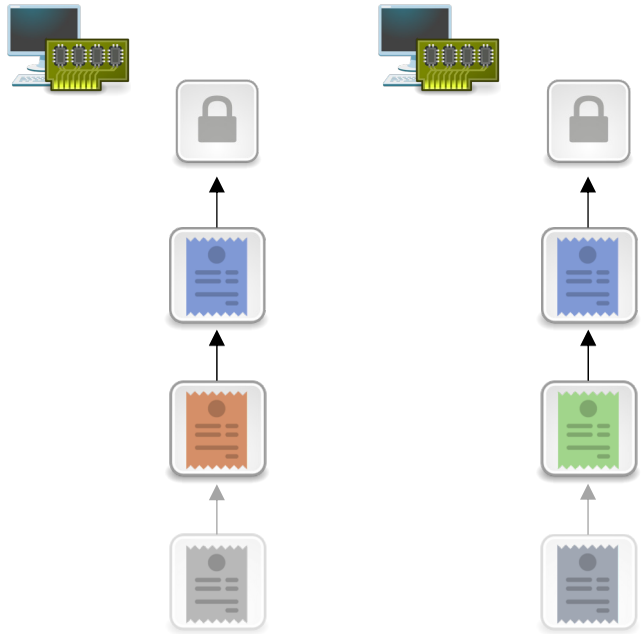


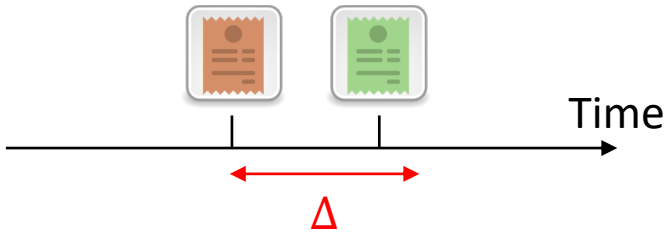


Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
$b^* \leftarrow GetLongestChain()$
$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
$txs \leftarrow GetPendingTxs()$
until new block arrives:
$\pi \leftarrow RandomNonce()$
$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
broadcast b_{new}

Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!

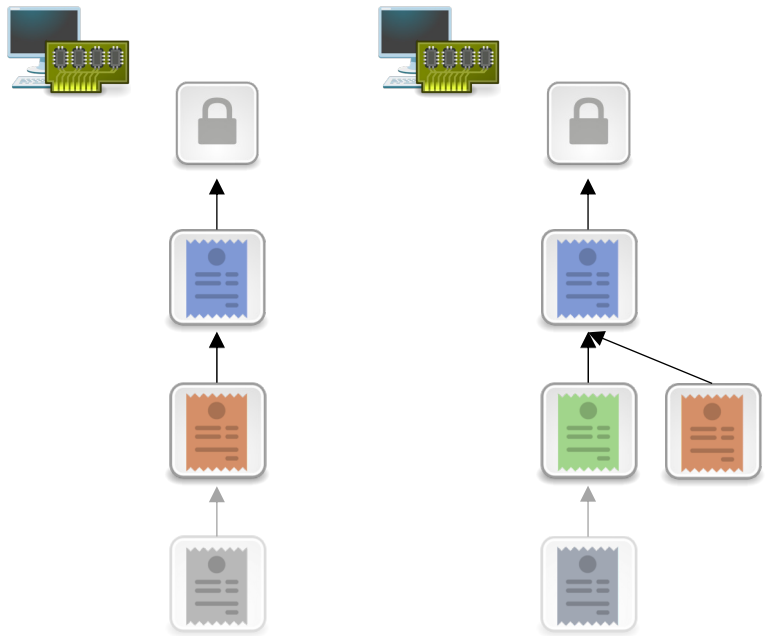


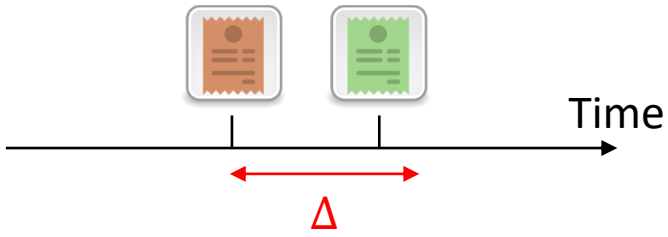


Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
$b^* \leftarrow GetLongestChain()$
$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
$txs \leftarrow GetPendingTxs()$
until new block arrives:
$\pi \leftarrow RandomNonce()$
$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
broadcast b_{new}

Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!





Security for $\Delta > 0$

Params: difficulty D, confirmation depth k

forever:

$b^* \leftarrow GetLongestChain()$

$LOG_p^t \leftarrow GetDeepTxs(b^*, k)$

$txs \leftarrow GetPendingTxs()$

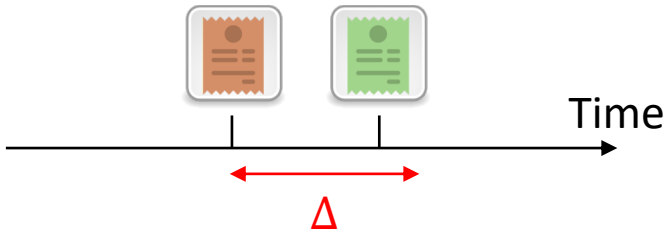
until new block arrives:

$\pi \leftarrow RandomNonce()$

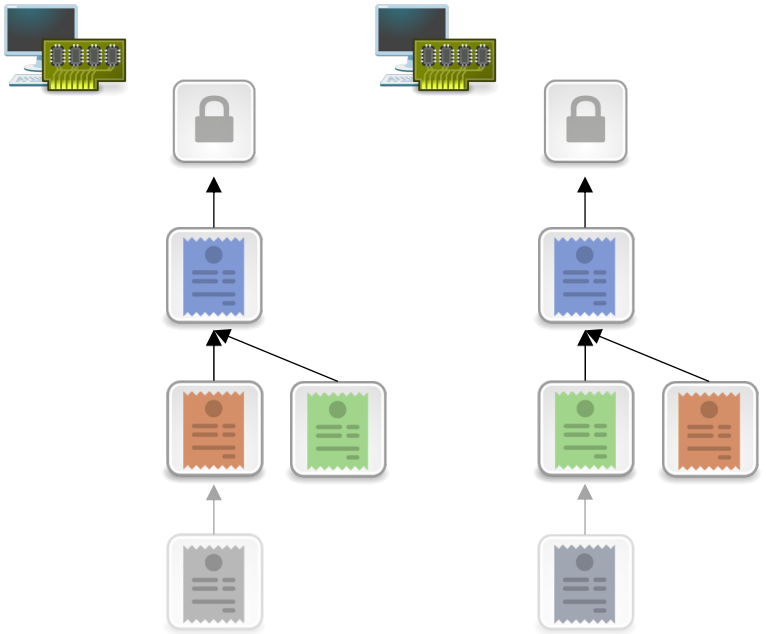
$b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$

if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:

broadcast b_{new}

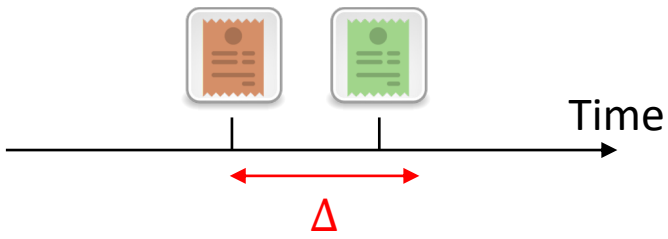


Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!

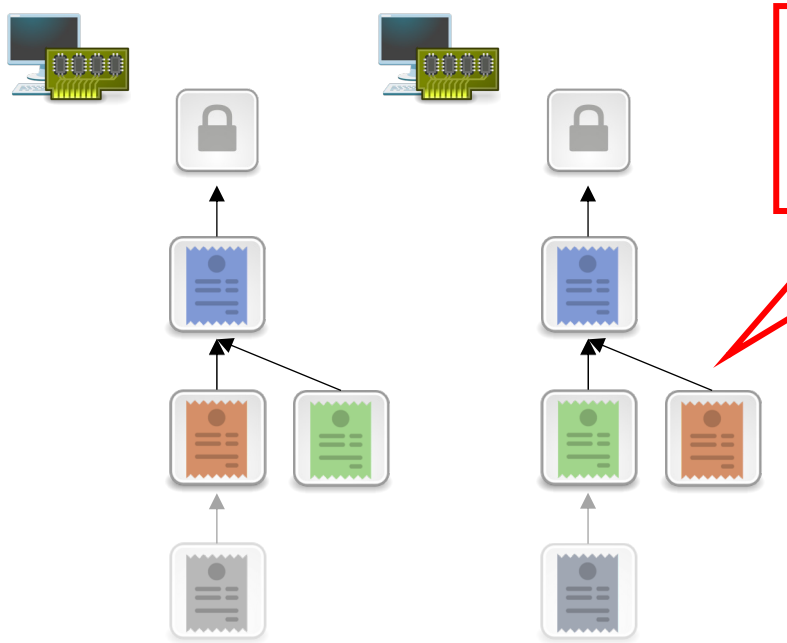


Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



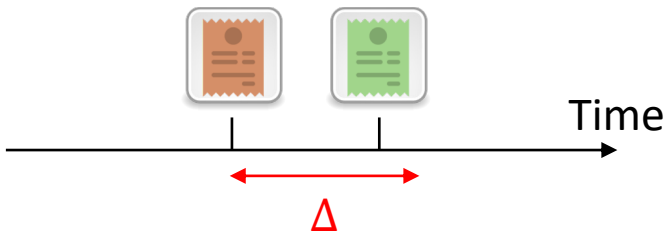
Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!



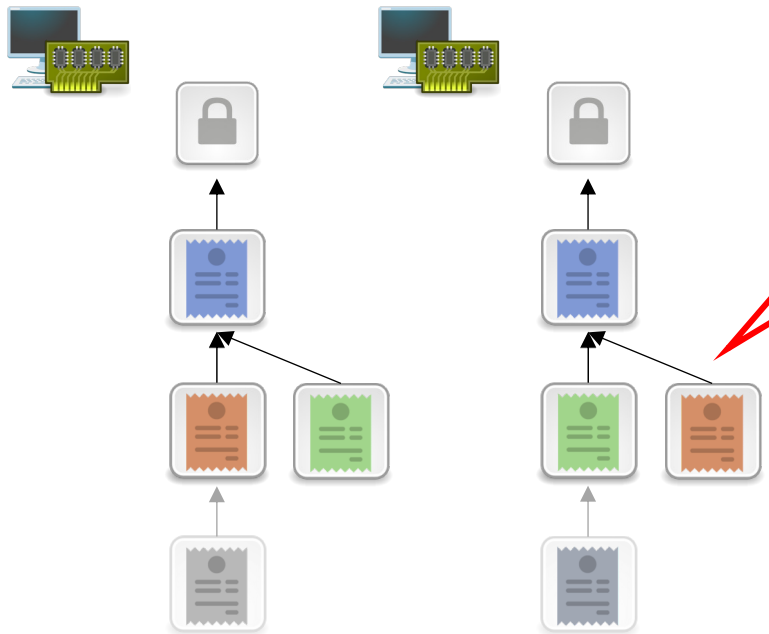
Forking! Multiple honest blocks on the same height!

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!

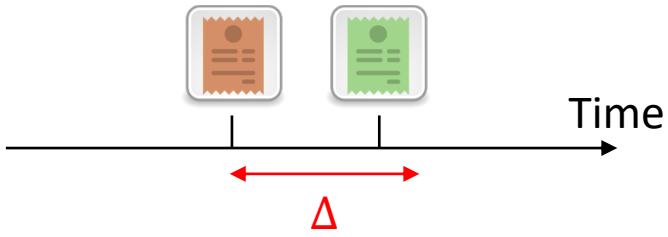


Forking! Multiple honest blocks on the same height!

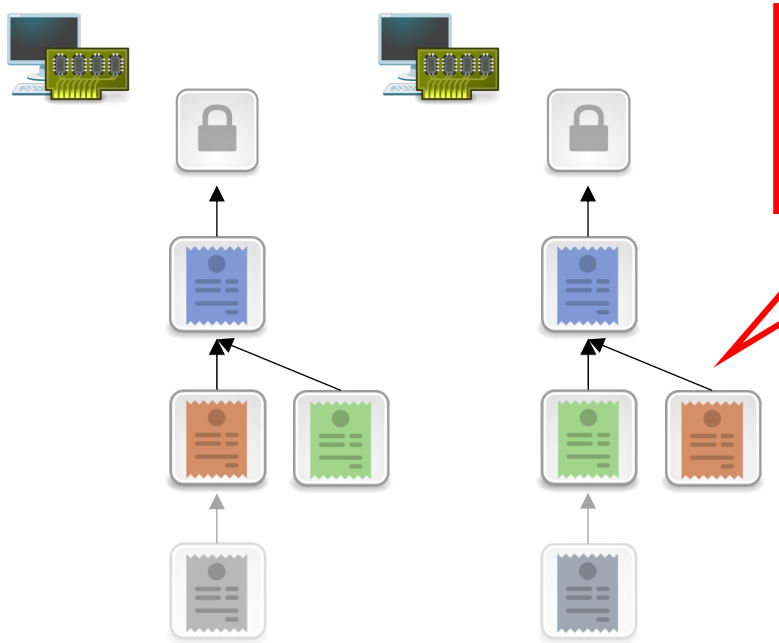
Bad, because it lowers the chain growth rate of the longest chain!

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!



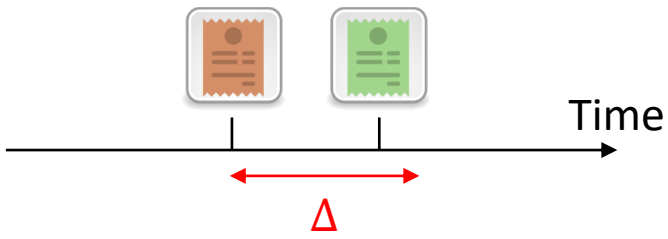
Forking! Multiple honest blocks on the same height!

Bad, because it lowers the chain growth rate of the longest chain!

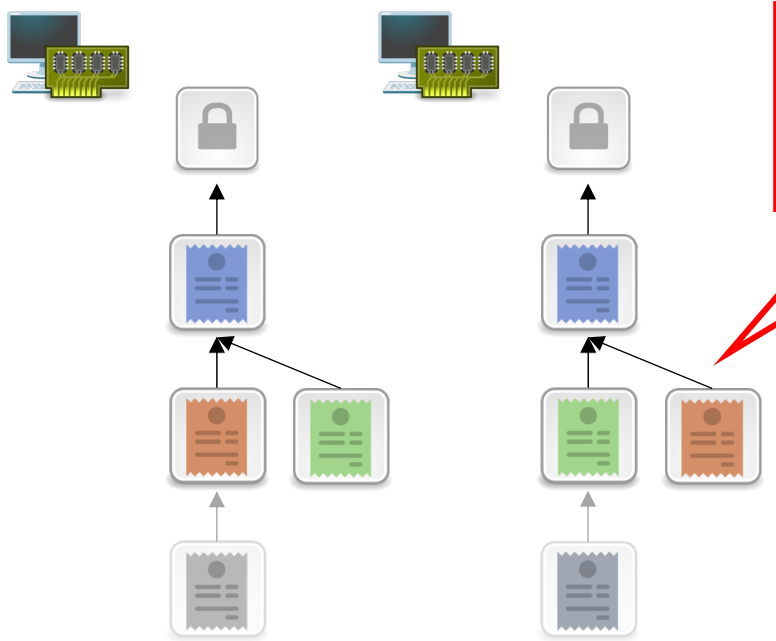
$$\lambda_{\text{growth}}^{\Delta=0} = \lambda_{\text{h}}$$

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!



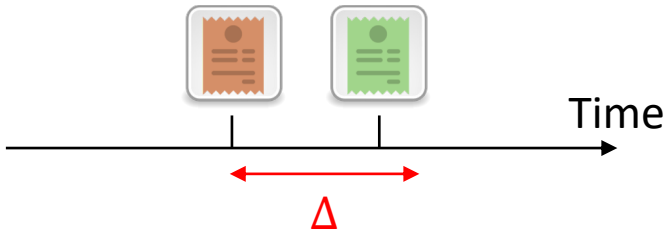
Forking! Multiple honest blocks on the same height!

Bad, because it lowers the chain growth rate of the longest chain!

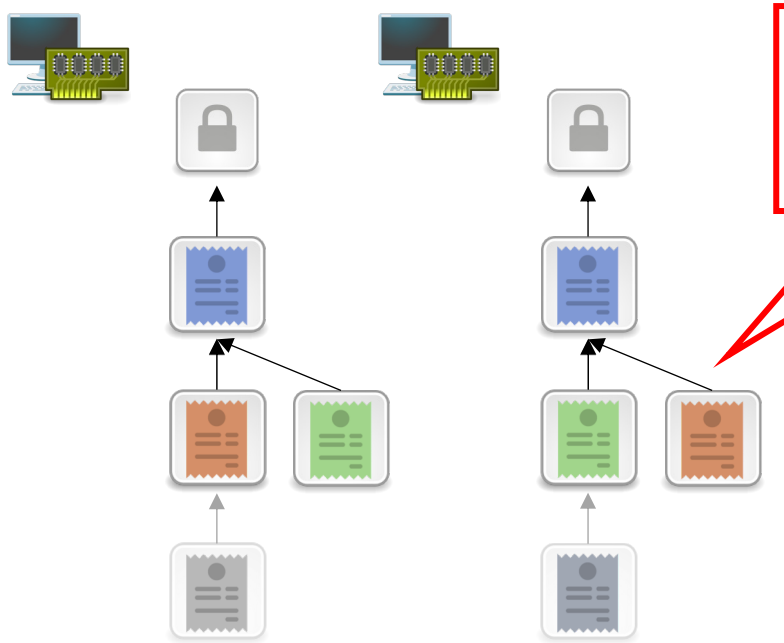
$\lambda_{\text{growth}}^{\Delta=0} = \lambda_{\text{h}}$

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!



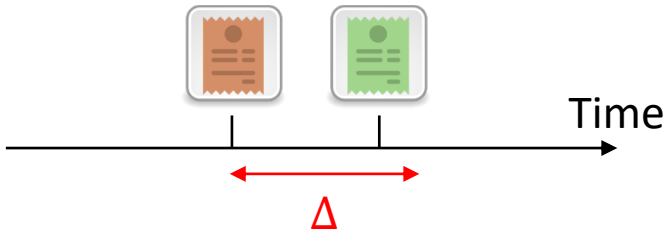
Forking! Multiple honest blocks on the same height!

Bad, because it lowers the chain growth rate of the longest chain!

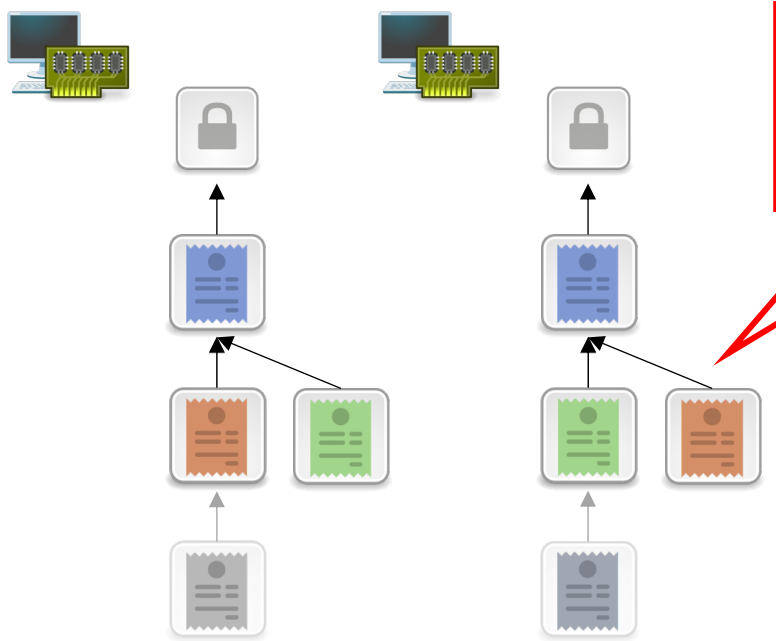
$$\lambda_{\text{growth}}^{\Delta=0} = \lambda_{\text{h}} \quad \Rightarrow \quad \lambda_{\text{growth}}^{\Delta>0} = \lambda_{\text{h}} \cdot \frac{1}{1 + \lambda_{\text{h}} \cdot \Delta}$$

Security for $\Delta > 0$

Params: difficulty D, confirmation depth k
forever:
  $b^* \leftarrow GetLongestChain()$
  $LOG_p^t \leftarrow GetDeepTxs(b^*, k)$
  $txs \leftarrow GetPendingTxs()$
  until new block arrives:
    $\pi \leftarrow RandomNonce()$
    $b_{\text{new}} \leftarrow NewBlock(b^*, txs, \pi)$
    if $Hash(b_{\text{new}}) \leq 2^{256} \cdot D$:
      broadcast b_{new}



Assume $\Delta > 0$.
→ Honest nodes do **not** see the same blocks!



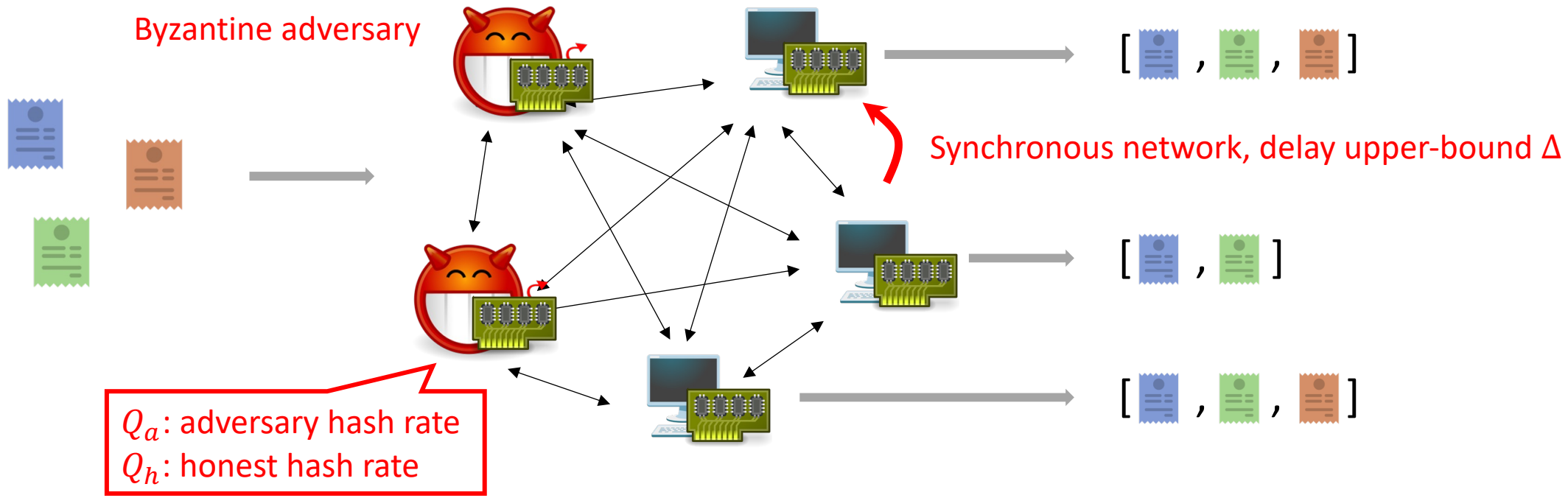
Forking! Multiple honest blocks on the same height!

Bad, because it lowers the chain growth rate of the longest chain!

Fraction of honest blocks not produced within Δ after an honest block

$$\lambda_{\text{growth}}^{\Delta=0} = \lambda_{\text{h}} \quad \Rightarrow \quad \lambda_{\text{growth}}^{\Delta>0} = \lambda_{\text{h}} \cdot \frac{1}{1 + \lambda_{\text{h}} \cdot \Delta}$$

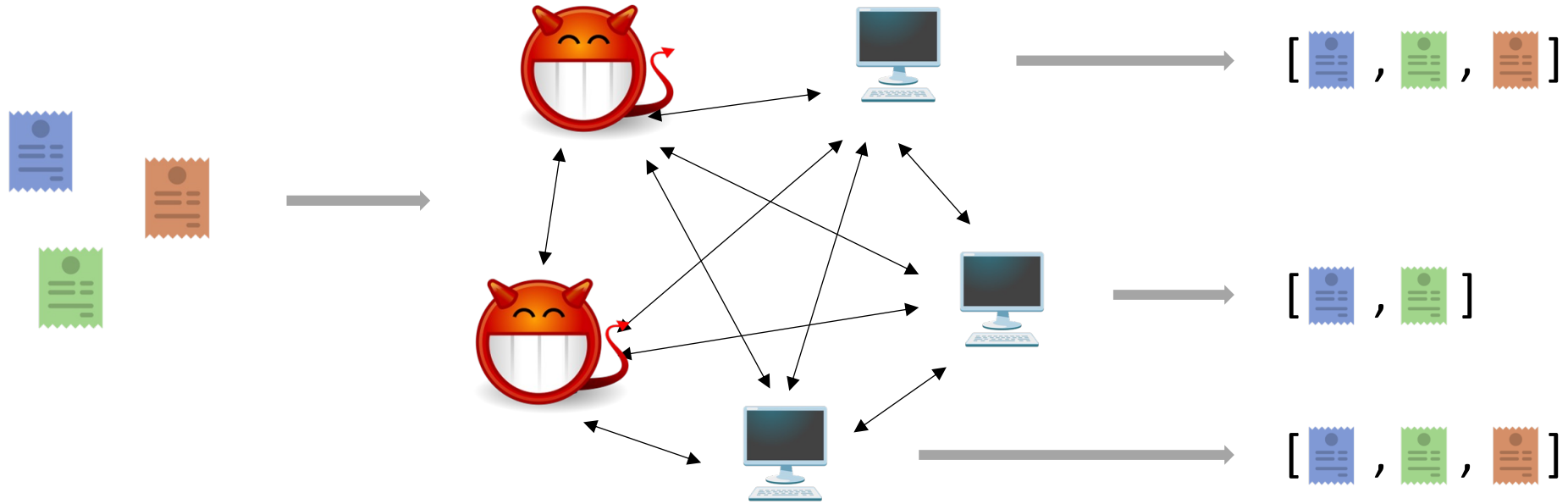
SMR Example 1: Proof-of-Work Nakamoto Consensus



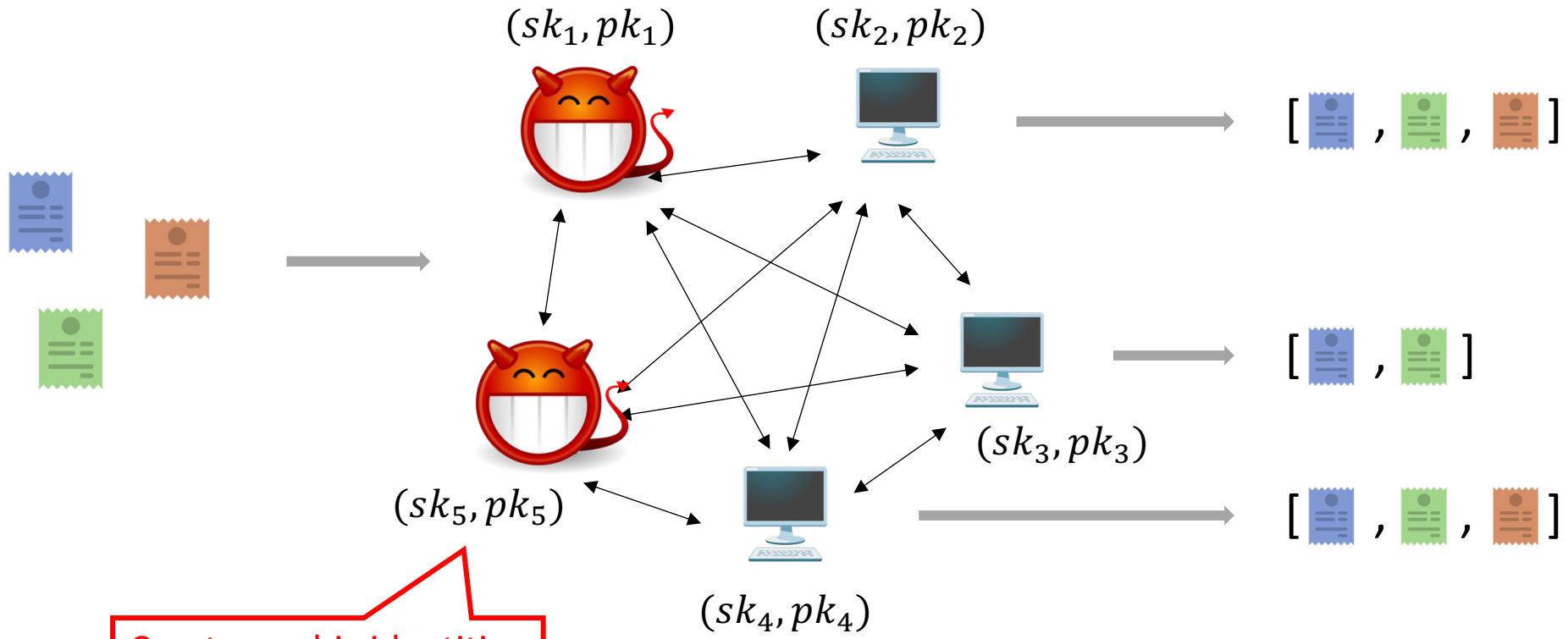
Theorem

Proof-of-work Nakamoto consensus with difficulty D is safe and live iff $Q_{\mathrm{a}} < Q_{\mathrm{h}} \cdot \frac{1}{1+Q_{\mathrm{h}} \cdot D \cdot \Delta}$, under synchrony, except with probability decaying exponentially in k.

SMR Example 2: Permissioned PBFT-Style Consensus

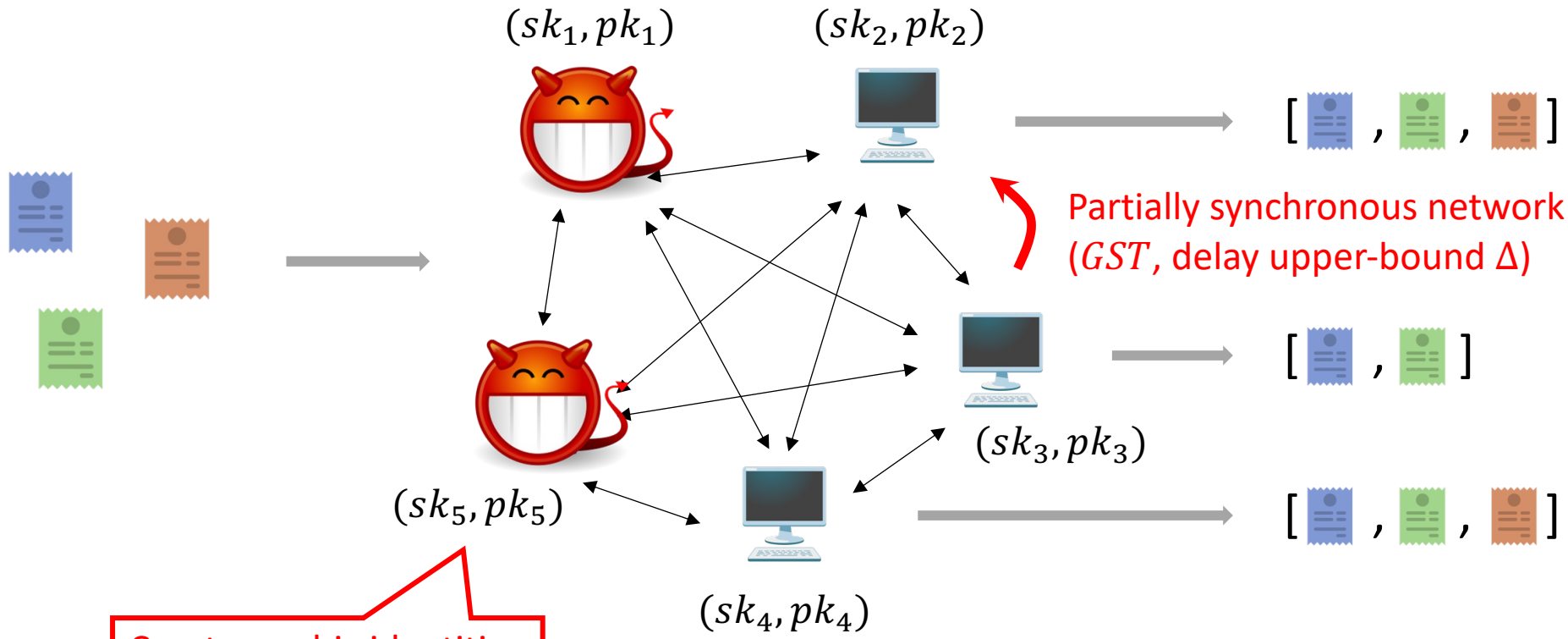


SMR Example 2: Permissioned PBFT-Style Consensus

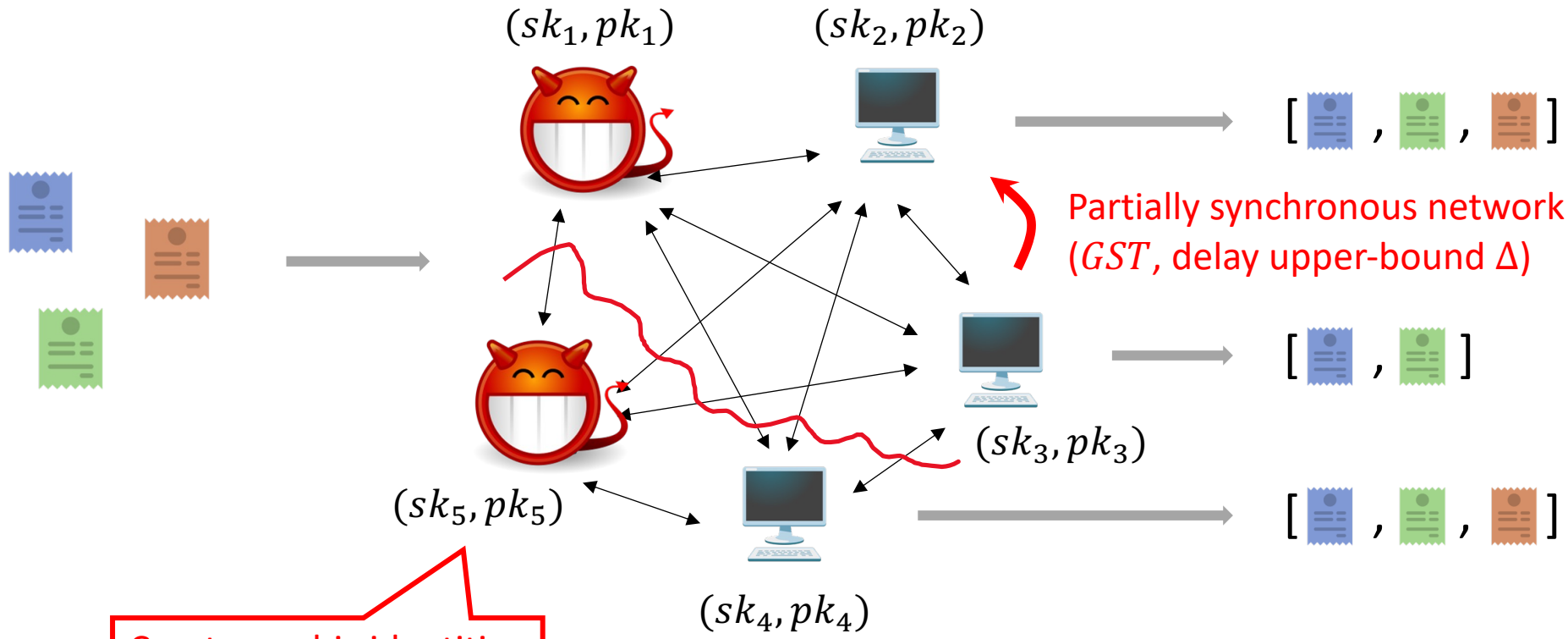


Cryptographic identities
n: nodes
f: adversary nodes
$n - f$: honest nodes

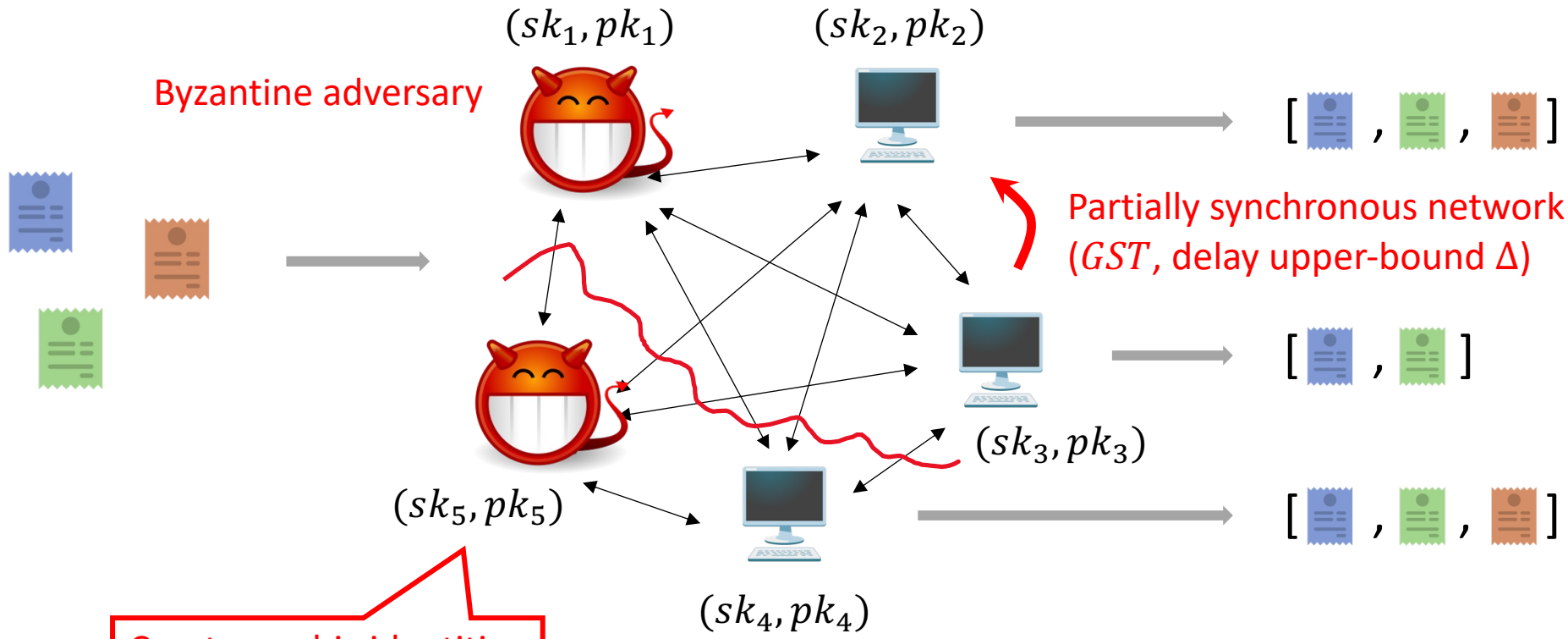
SMR Example 2: Permissioned PBFT-Style Consensus



SMR Example 2: Permissioned PBFT-Style Consensus

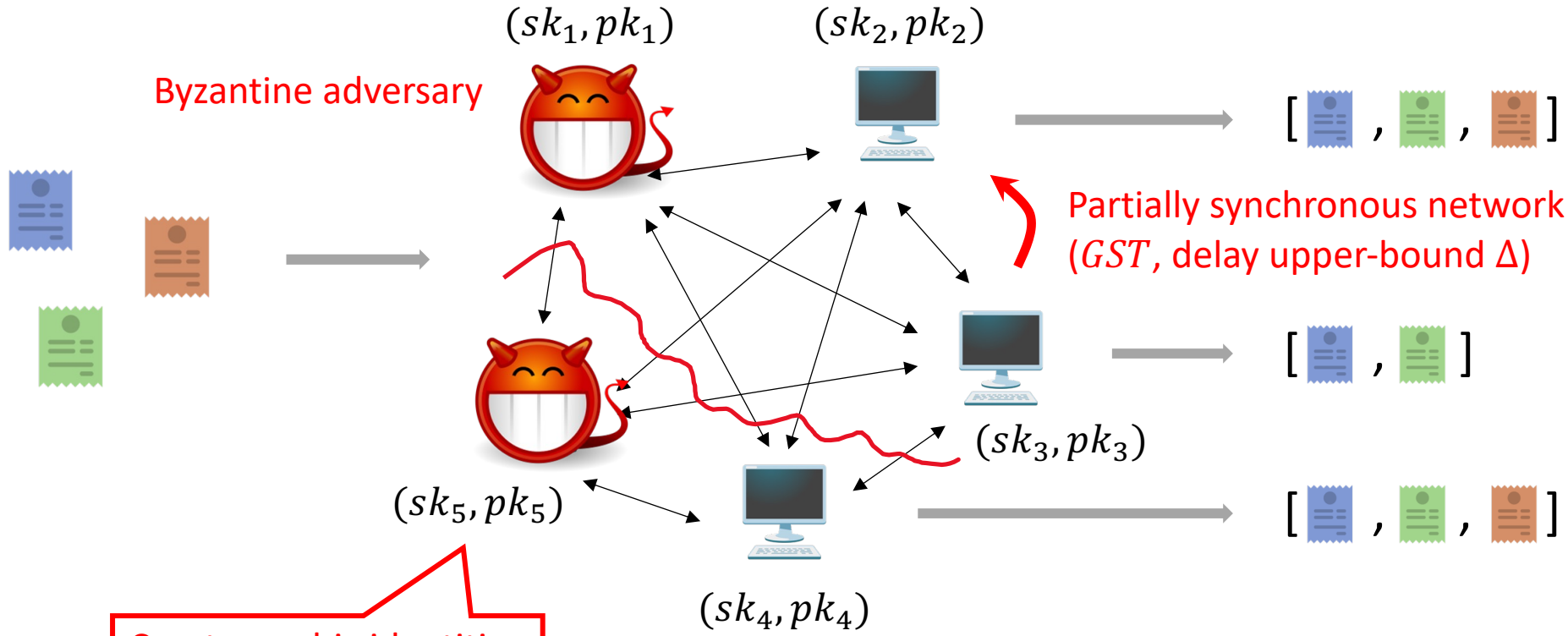


SMR Example 2: Permissioned PBFT-Style Consensus



SMR Example 2: Permissioned PBFT-Style Consensus

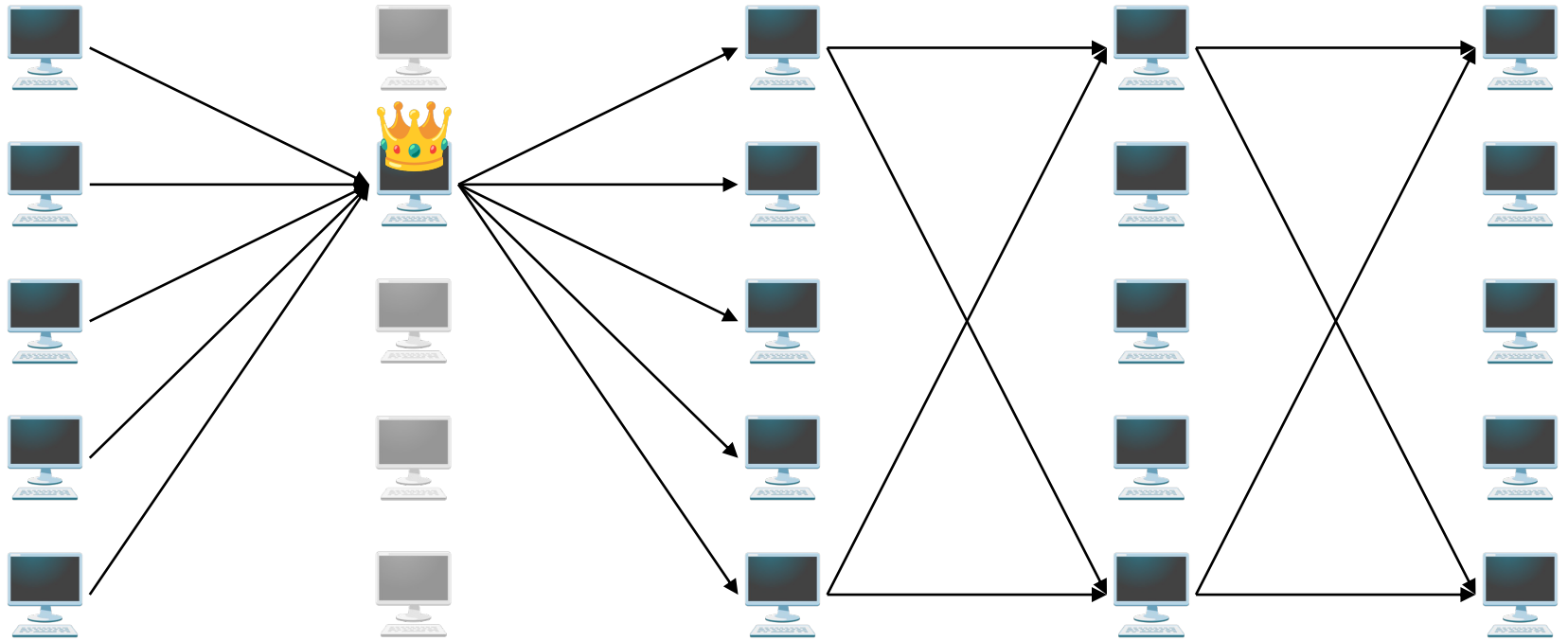
Pre-dates blockchains by 10+ years



Cryptographic identities
n: nodes
f: adversary nodes
$n - f$: honest nodes

PBFT-Style Consensus

Proceeds in views, each with a proposal by a designated leader followed by two rounds of votes (with a quorum of $n - f$).

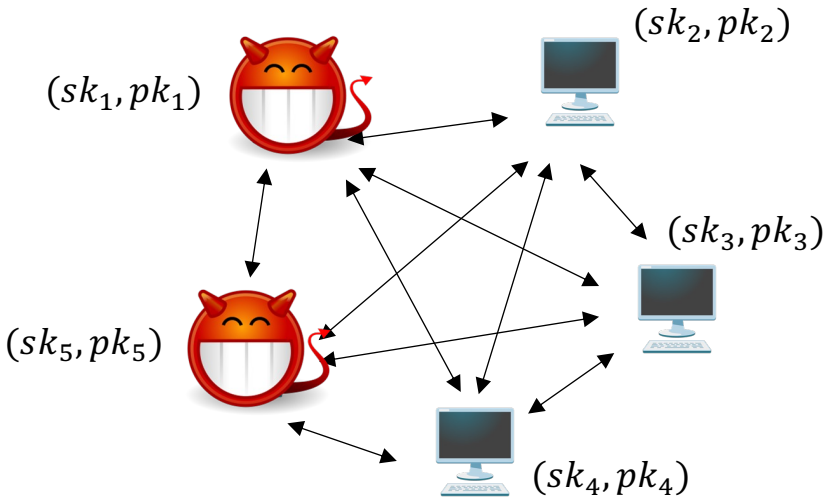


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



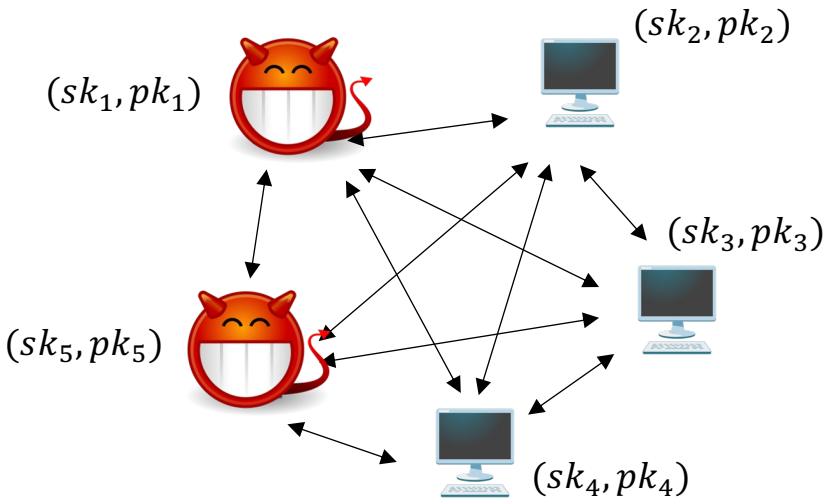
PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

"Catch-up", view change



PBFT-Style Consensus

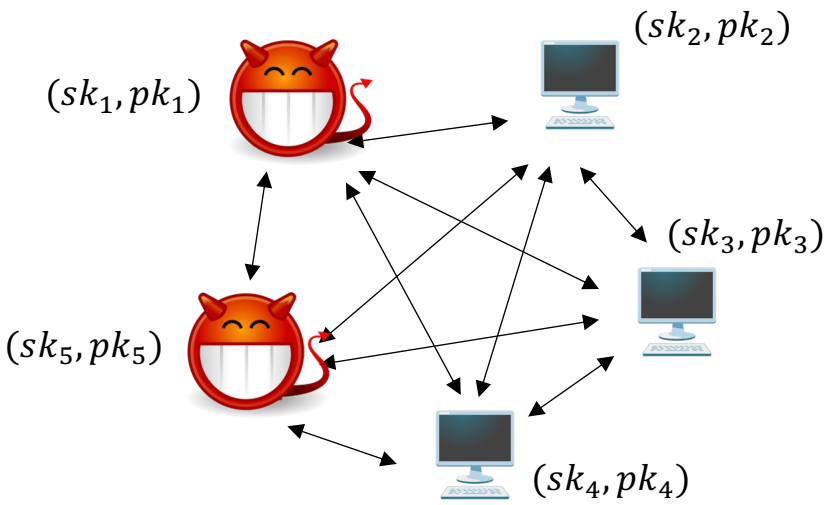
$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

Leader proposal

"Catch-up", view change



PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

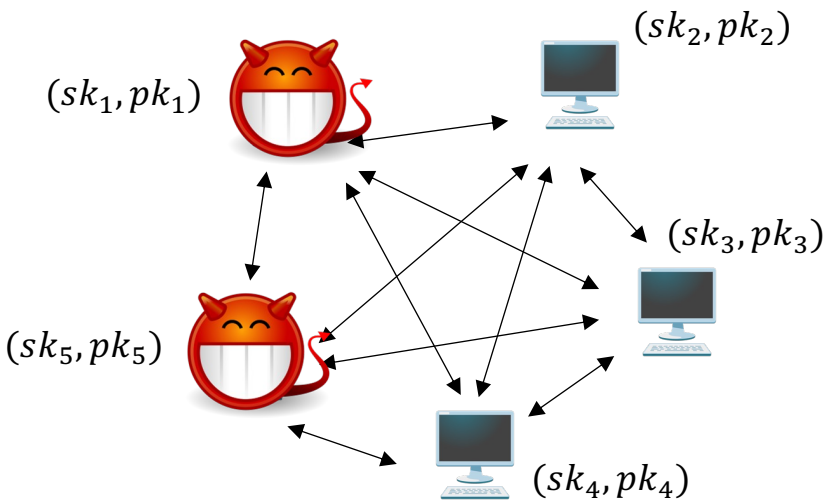
Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

Leader proposal

"Works for me", vote 1

"Catch-up", view change



PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

Example: $n = 7, f = 2 \rightarrow$ quorum 5

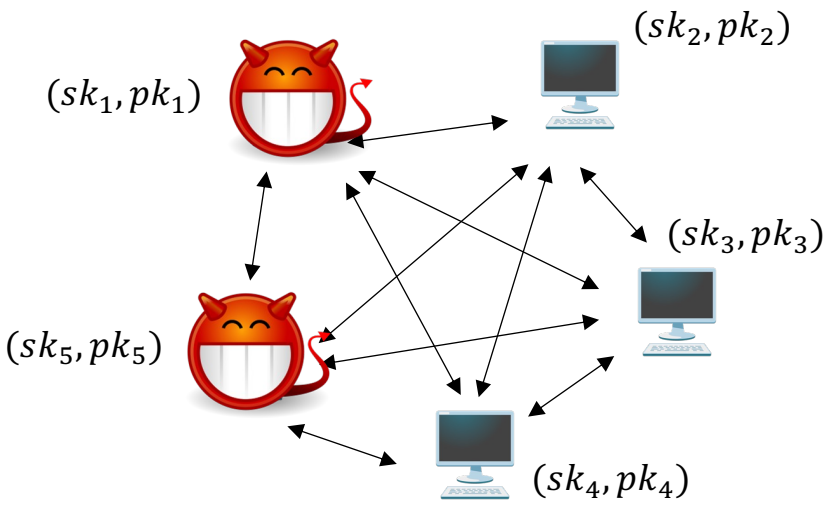
Genesis block

Leader proposal

"Works for me", vote 1

"Let's do it", vote 2

"Catch-up", view change

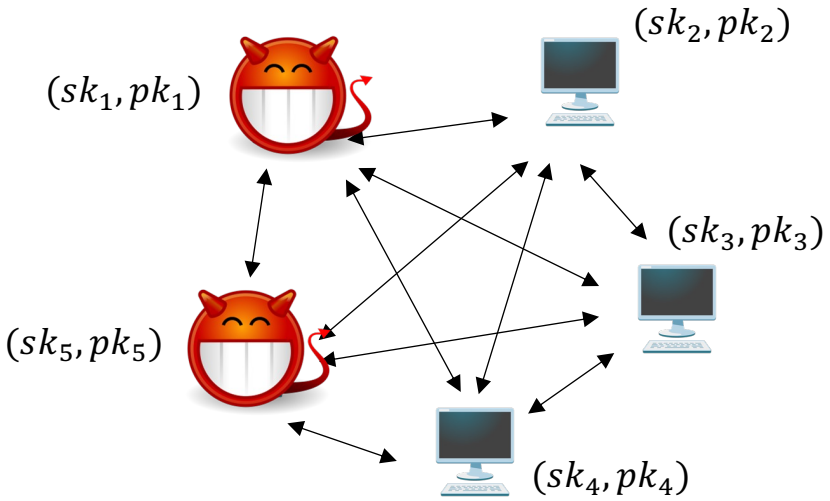


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

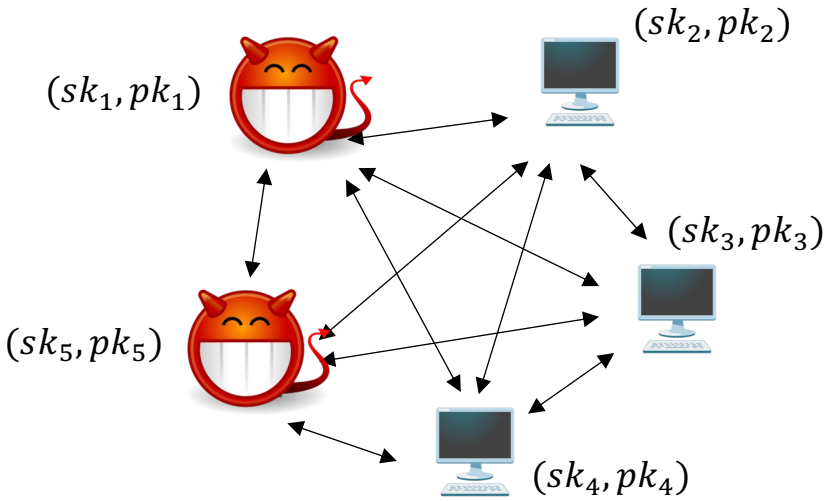


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

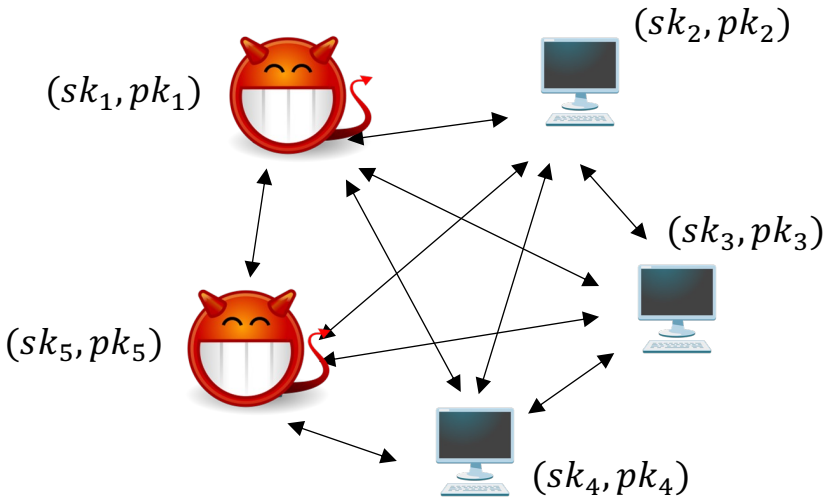


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

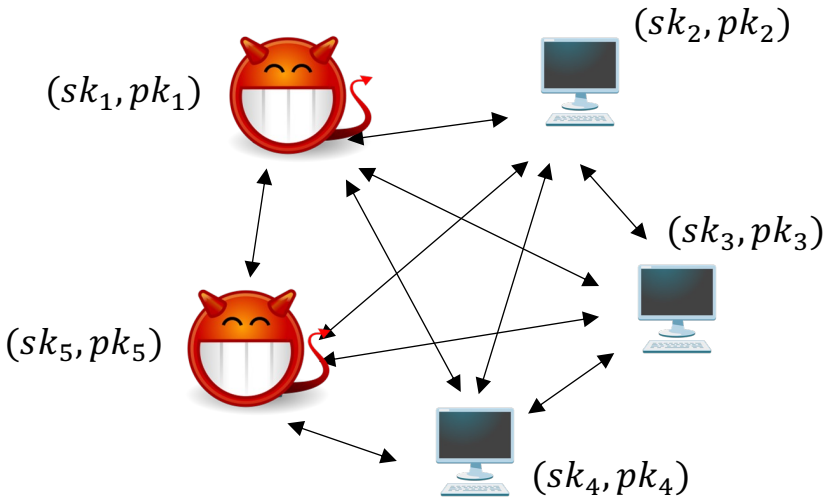


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

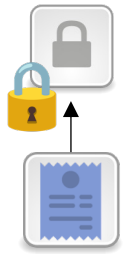
$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



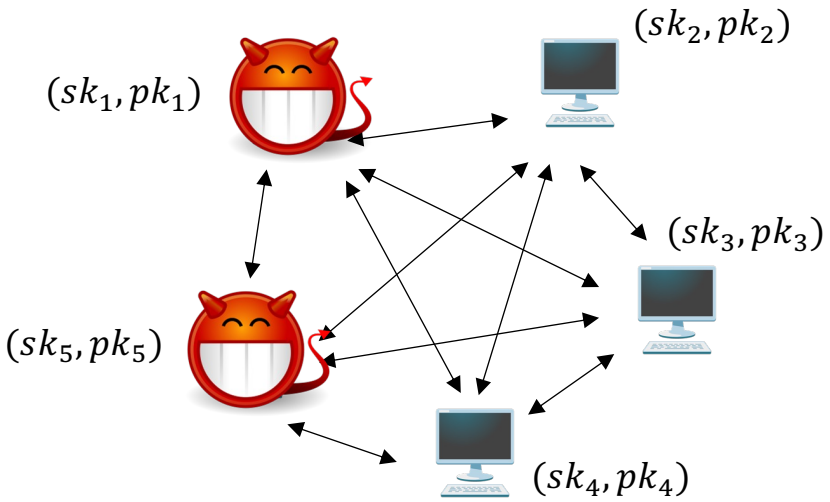
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

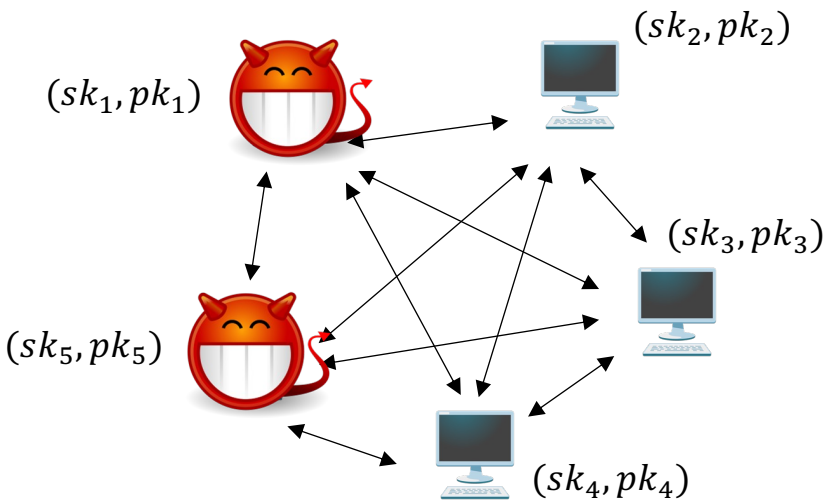


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



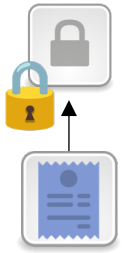
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Example: $n = 7, f = 2 \rightarrow$ quorum 5

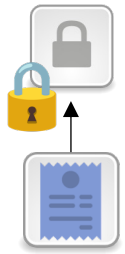
Genesis block



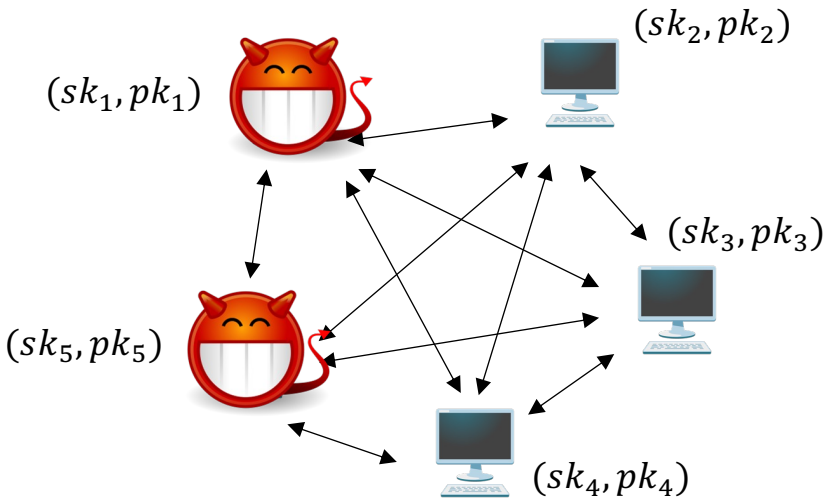
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

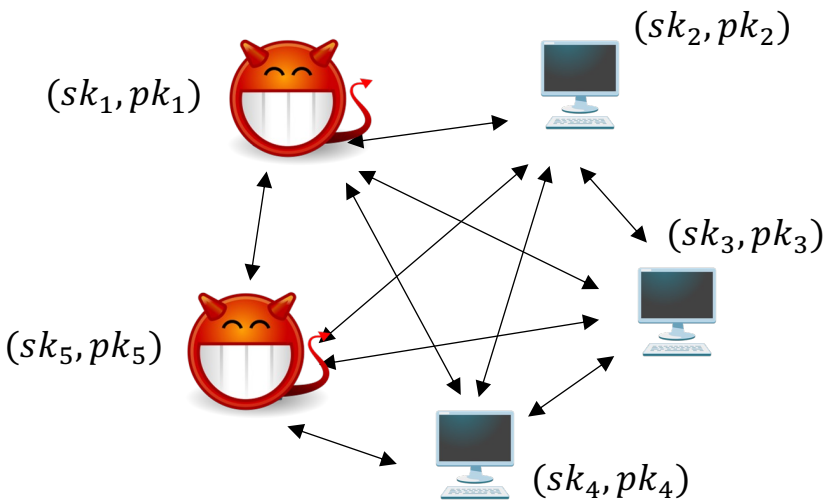


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



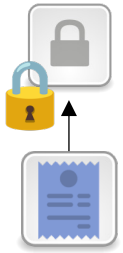
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Example: $n = 7, f = 2 \rightarrow$ quorum 5

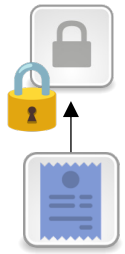
Genesis block



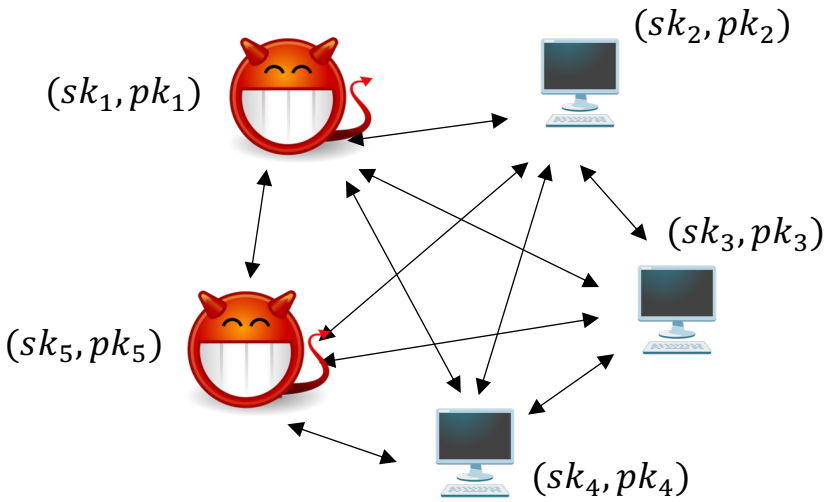
PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block



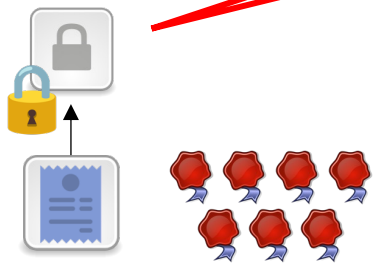
$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



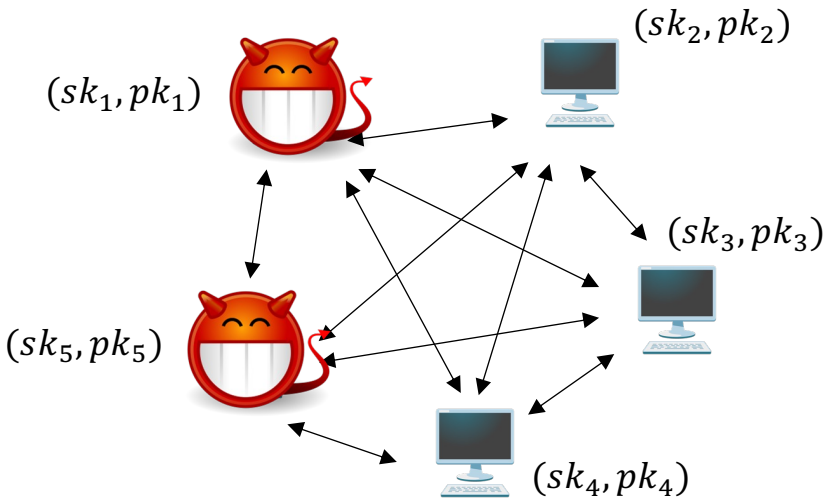
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

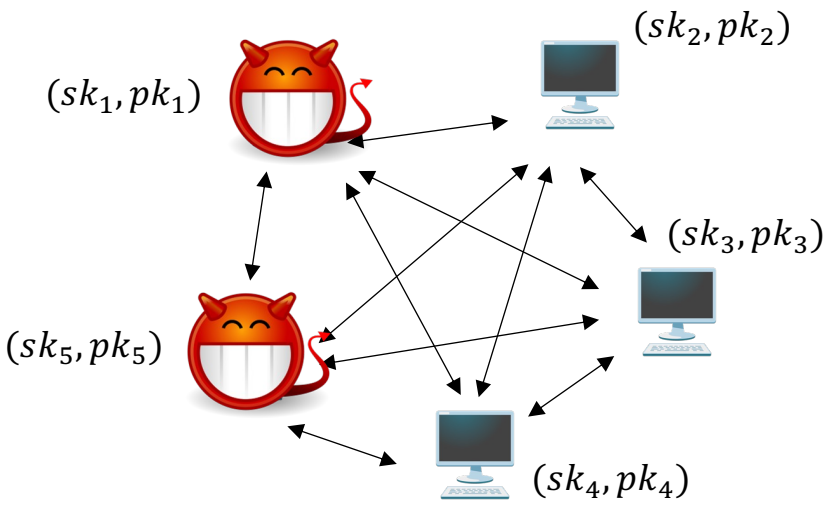


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



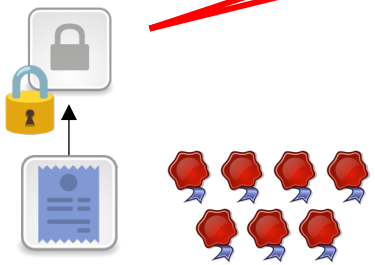
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Example: $n = 7, f = 2 \rightarrow$ quorum 5

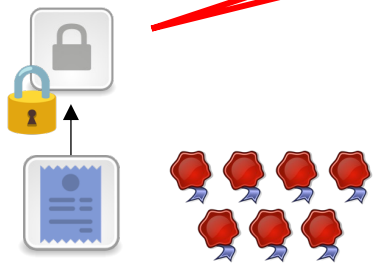
Genesis block



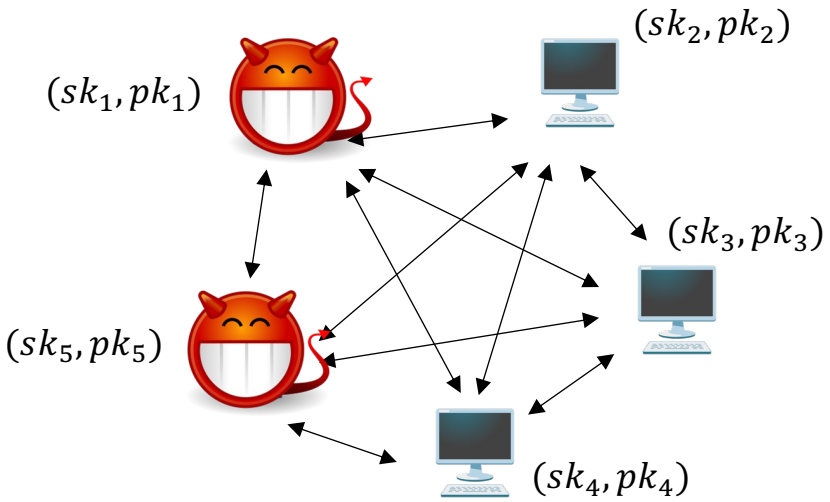
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

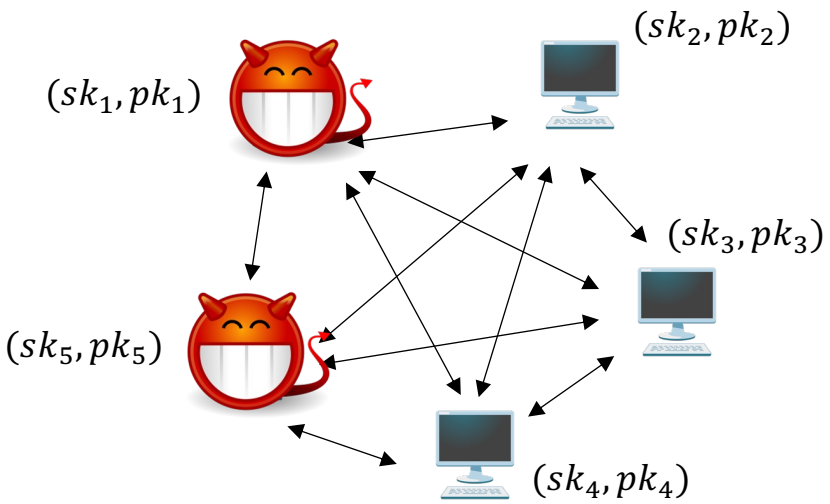


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



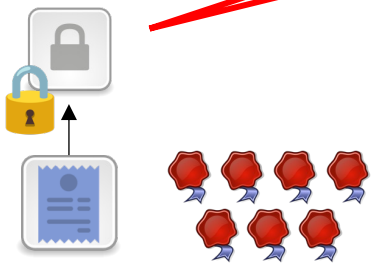
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



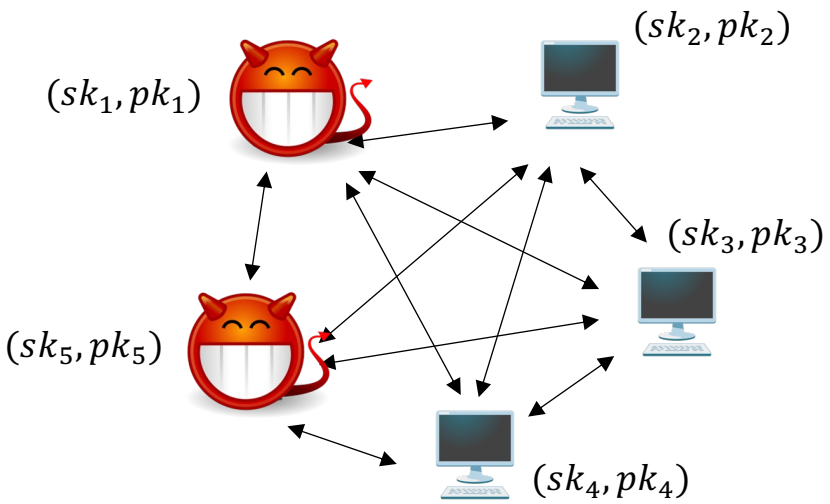
Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block



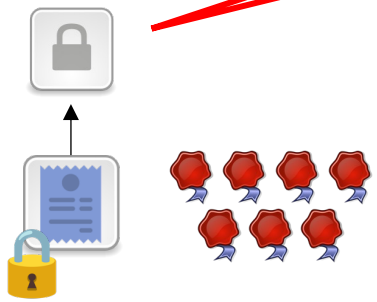
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Example: $n = 7, f = 2 \rightarrow$ quorum 5

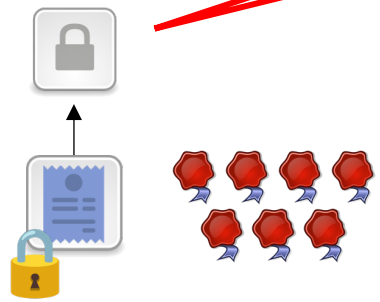
Genesis block



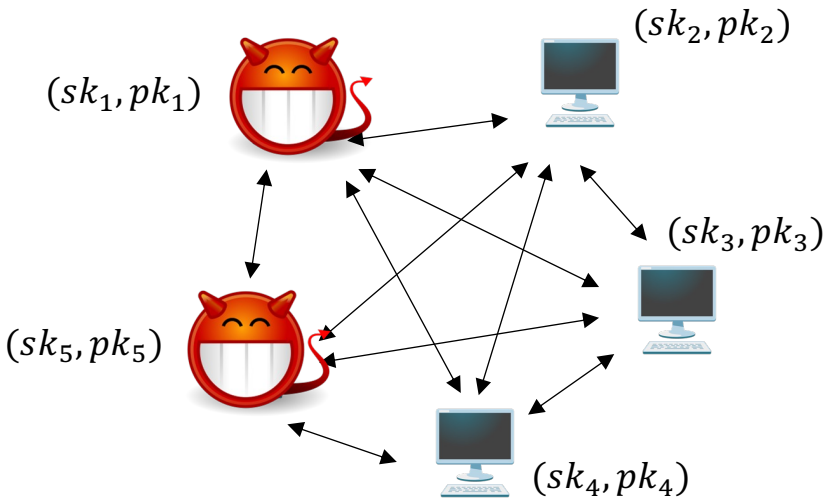
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block



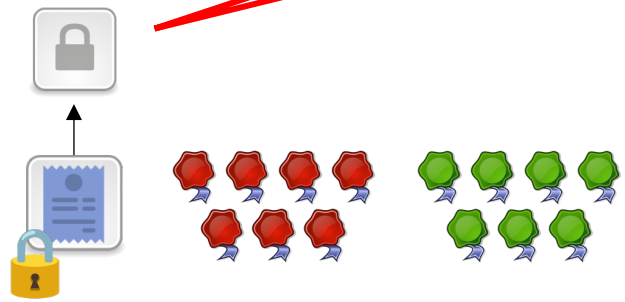
$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



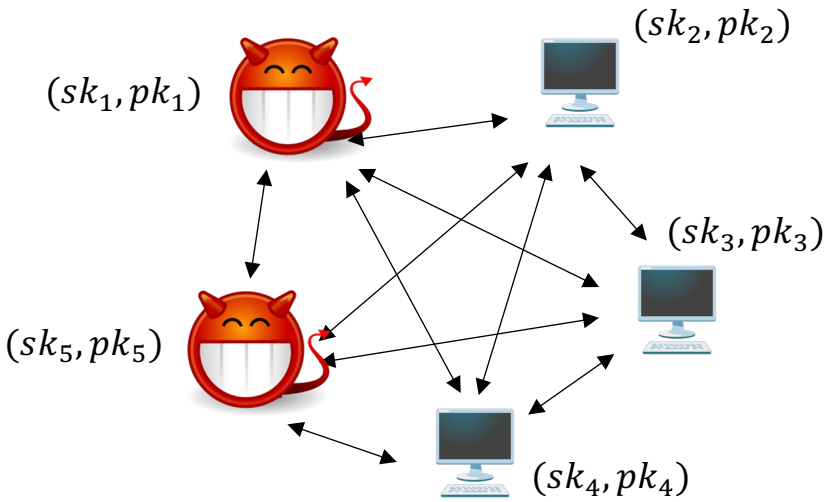
PBFT-Style Consensus

Example: $n = 7, f = 2 \rightarrow$ quorum 5

Genesis block

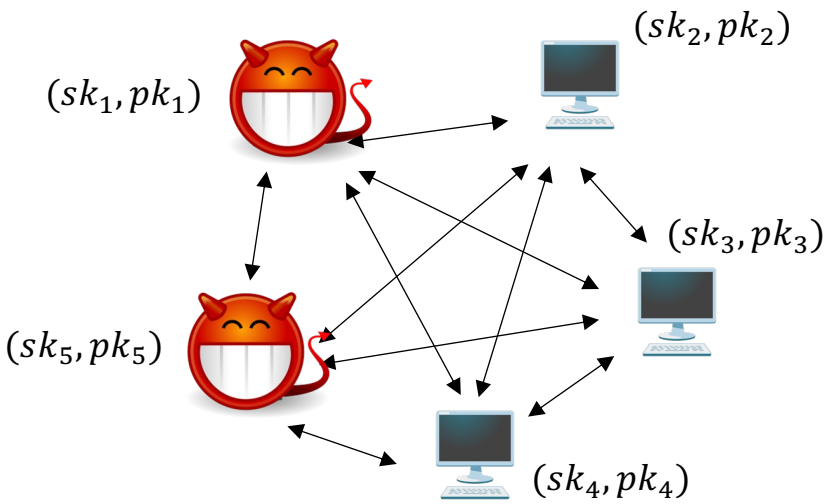


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



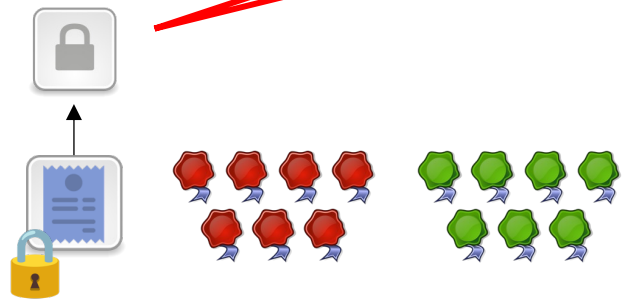
PBFT-Style Consensus

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Example: $n = 7, f = 2 \rightarrow$ quorum 5

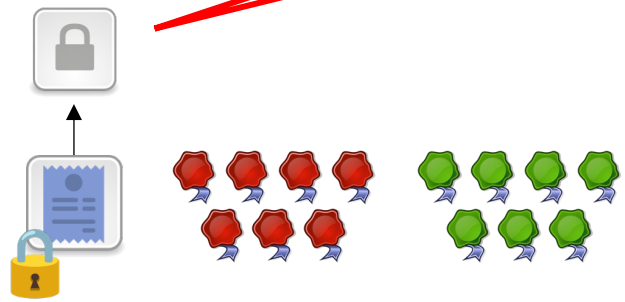
Genesis block



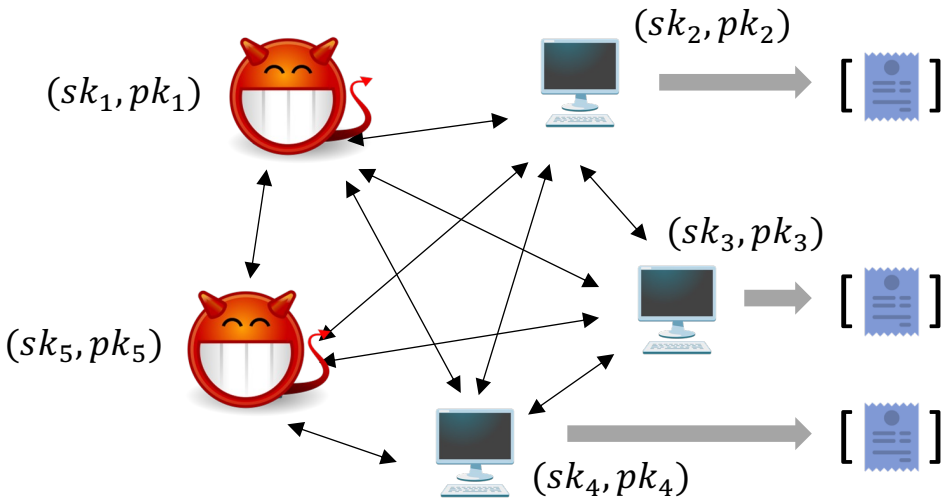
PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

Genesis block

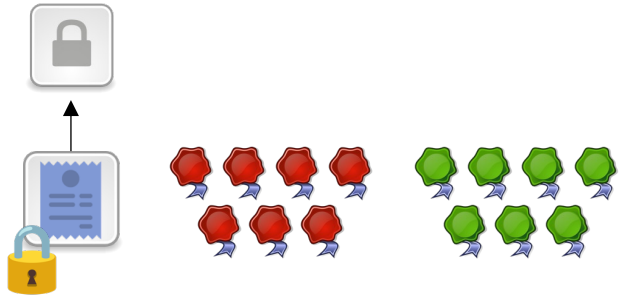


$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

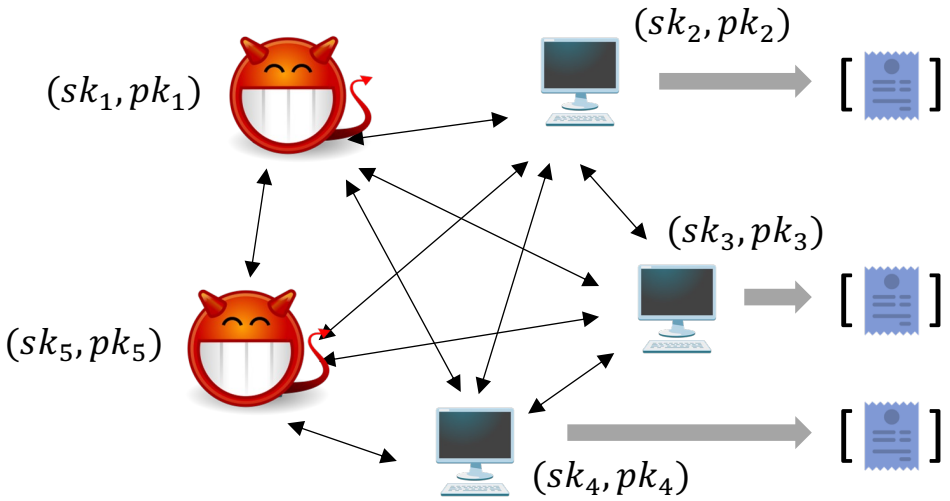


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5



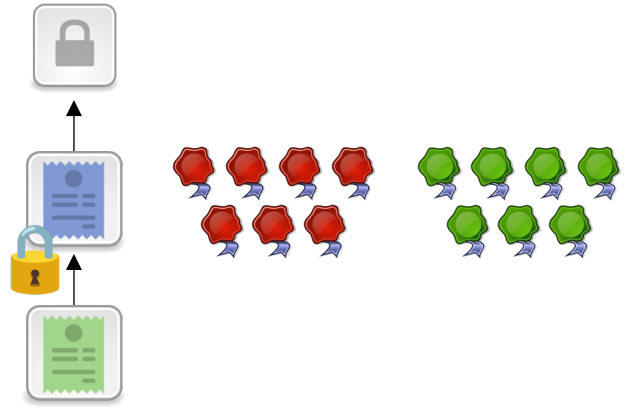
$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

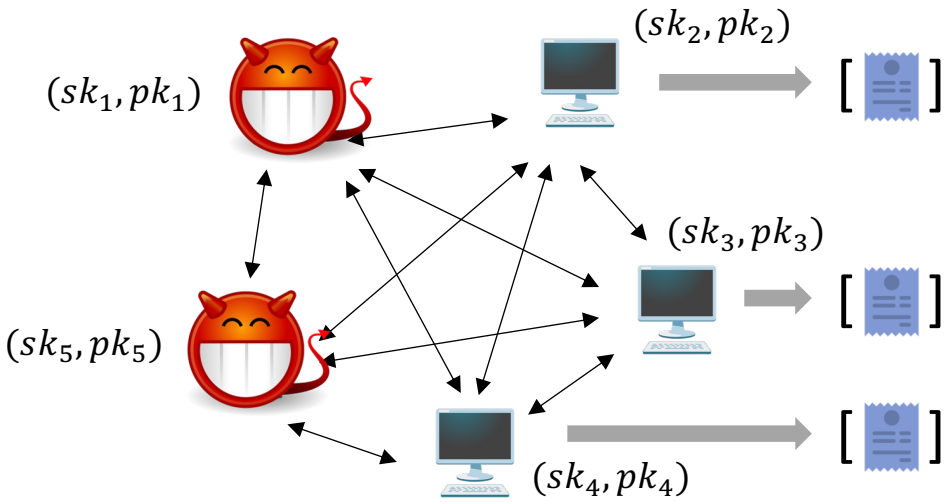


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

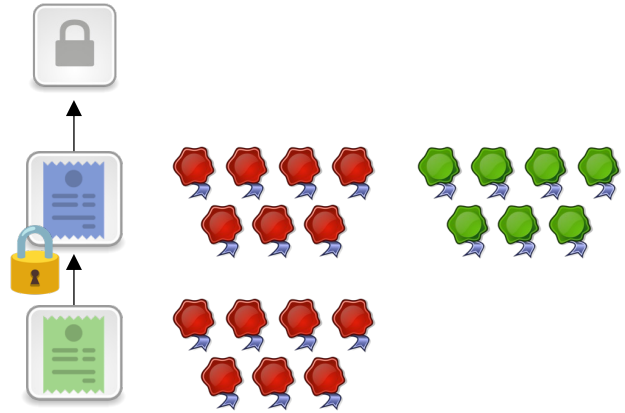


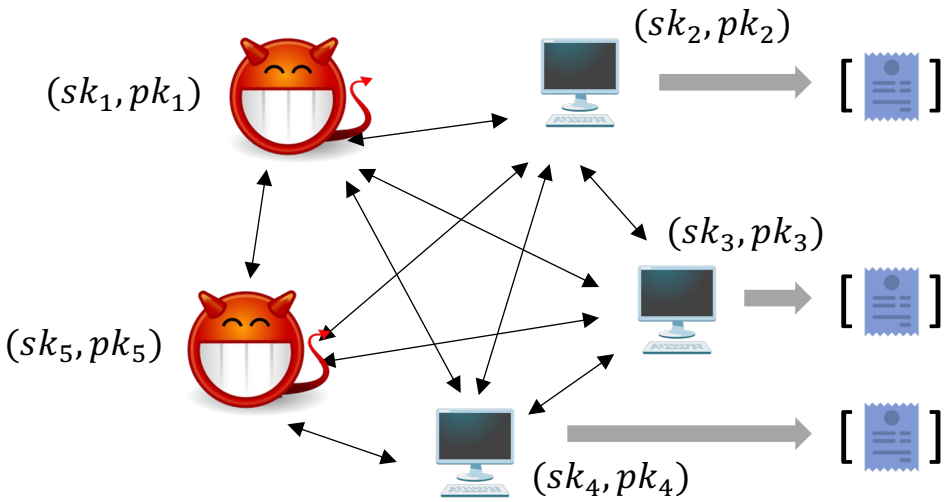


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

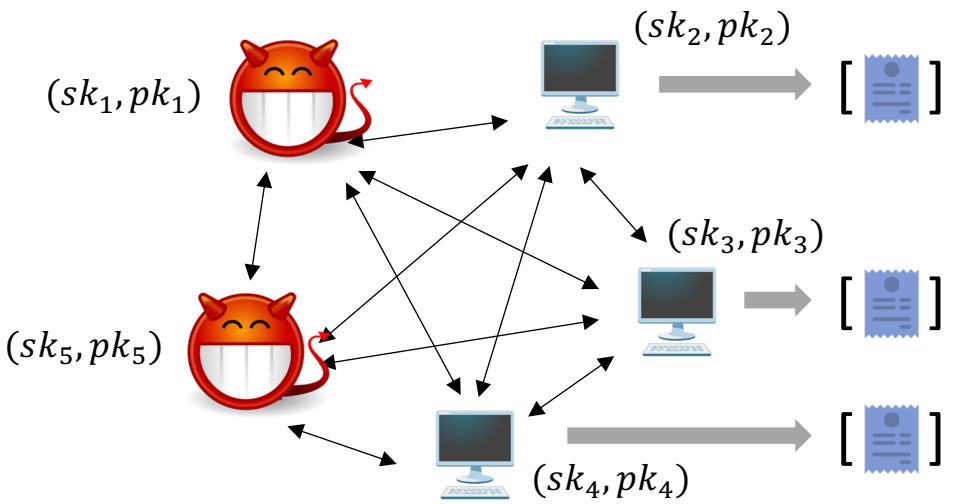


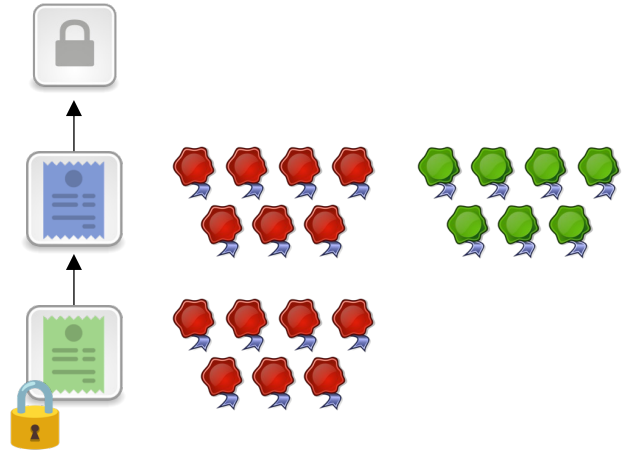


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

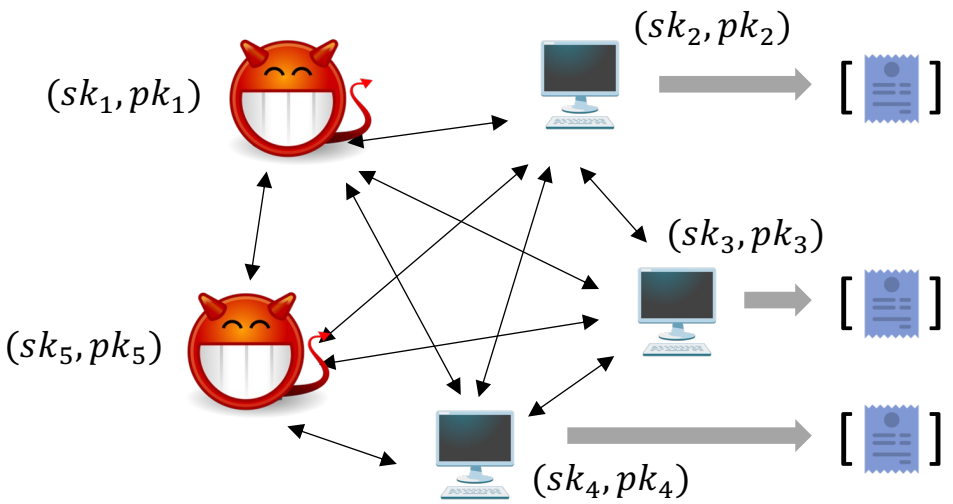


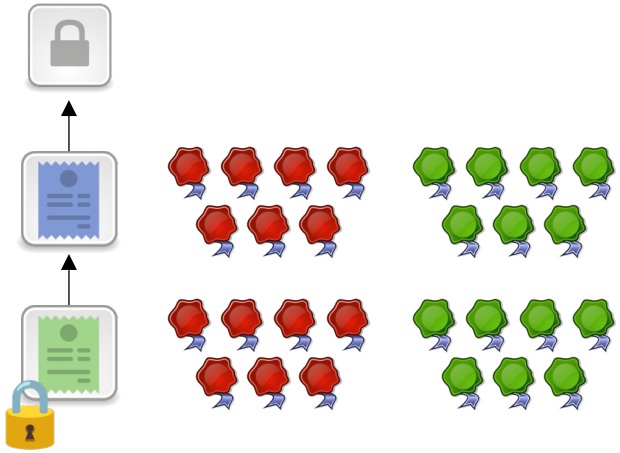


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

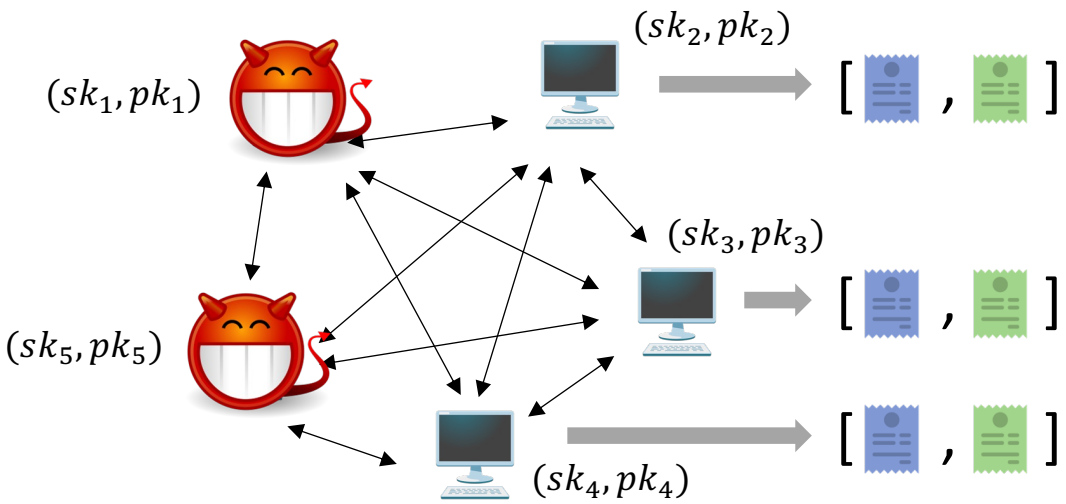


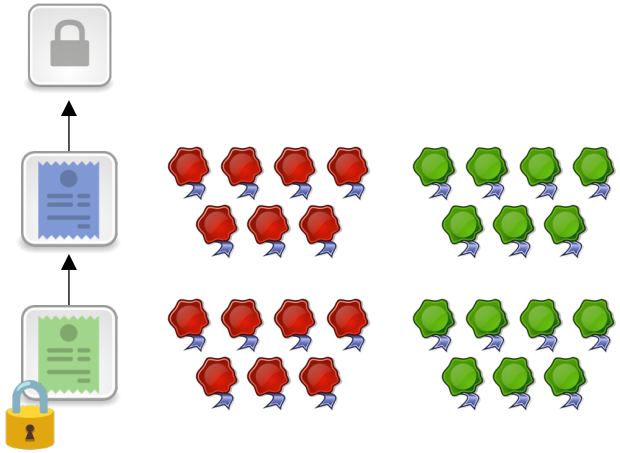


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

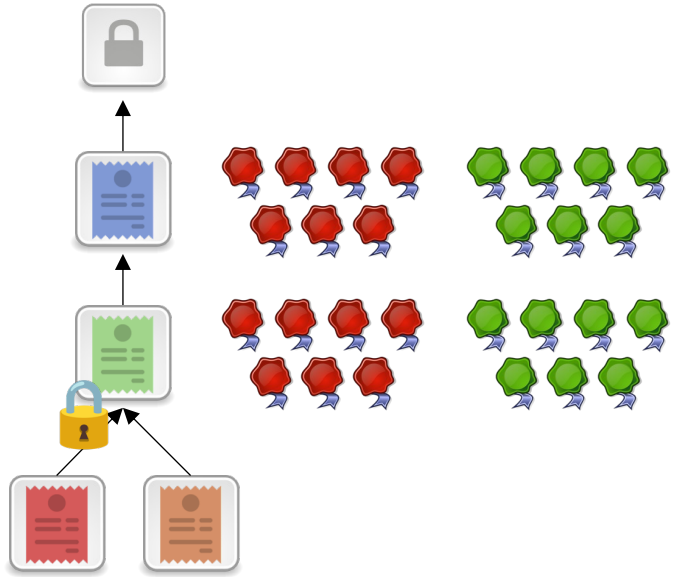


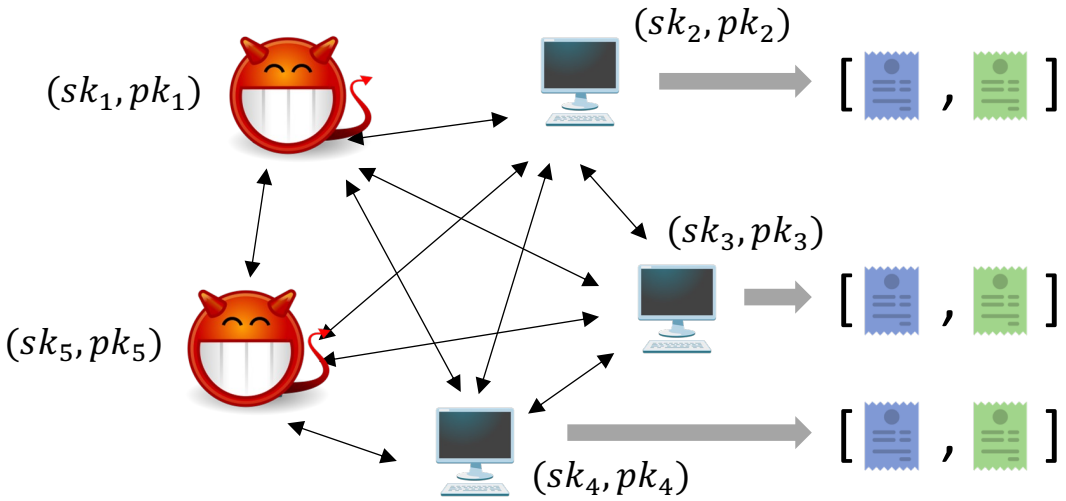


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

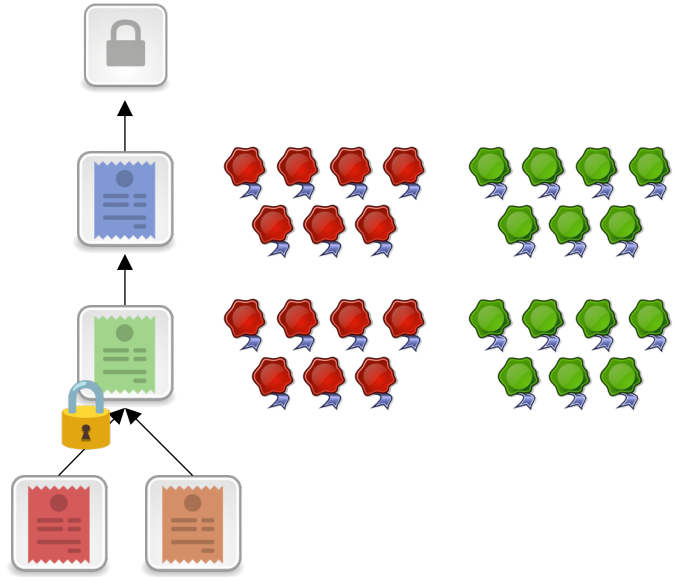


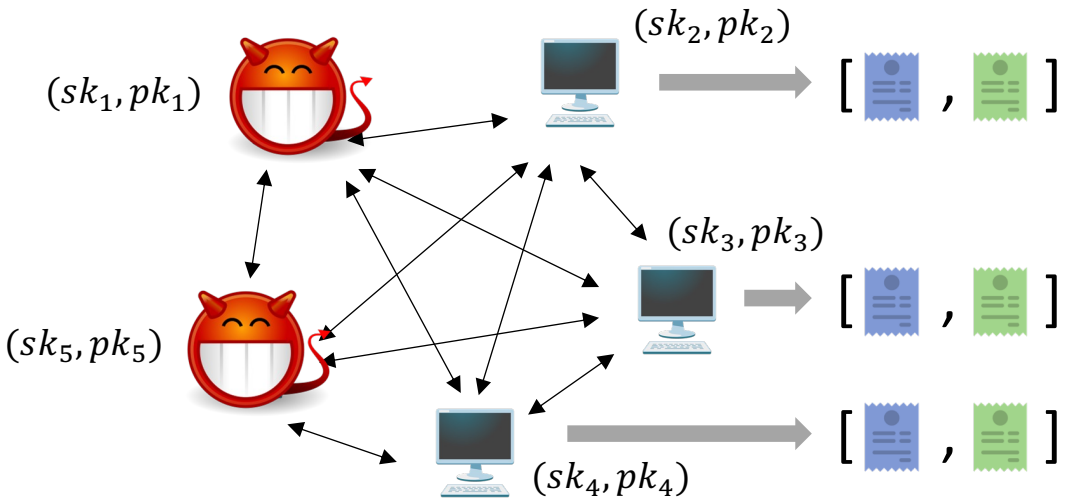


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

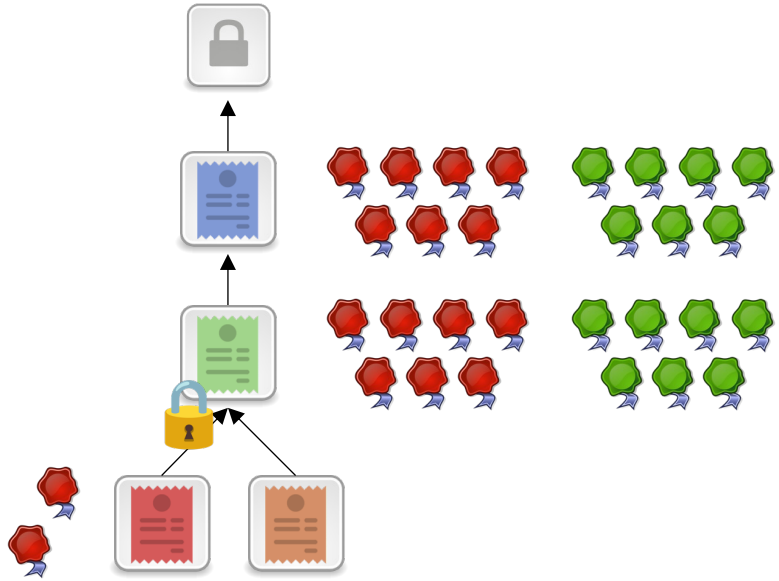


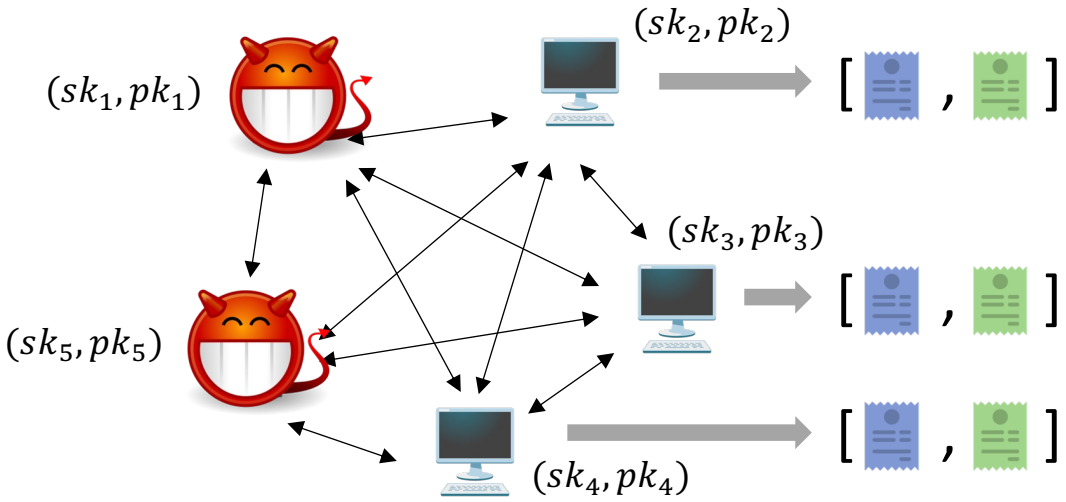


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

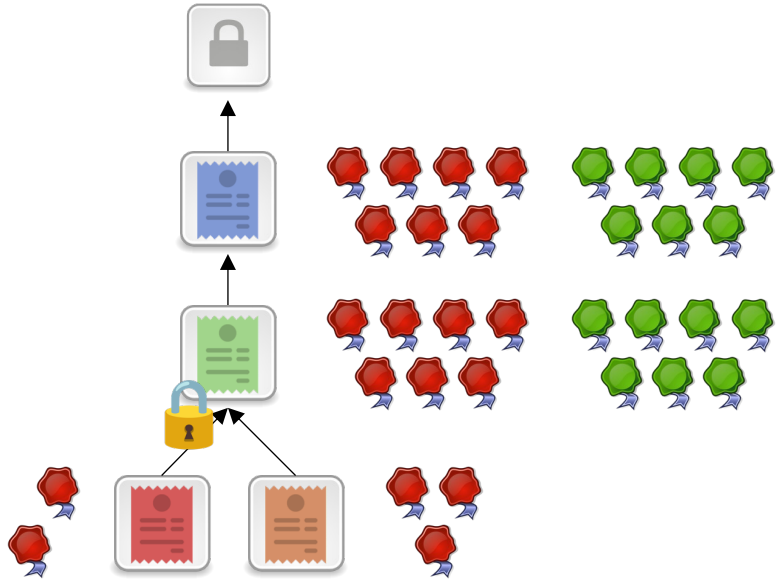


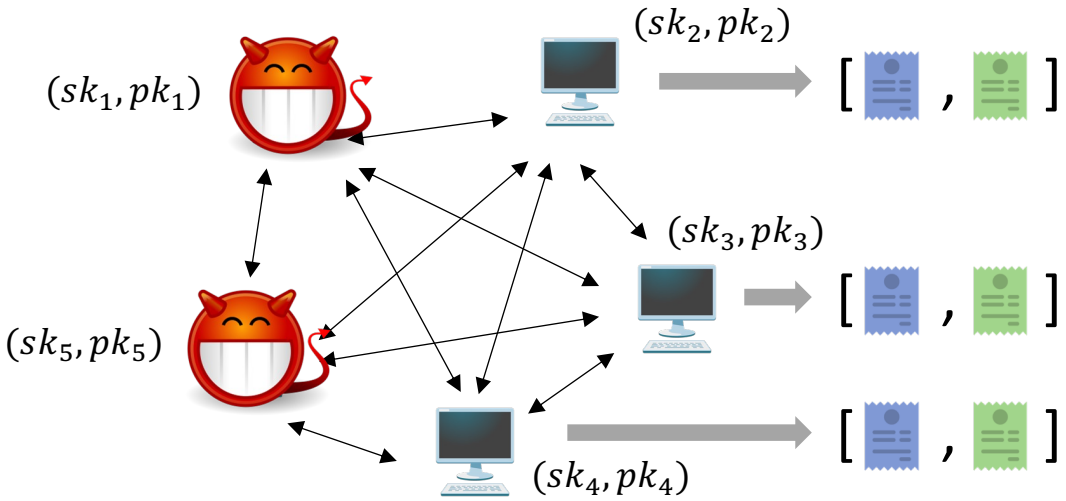


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

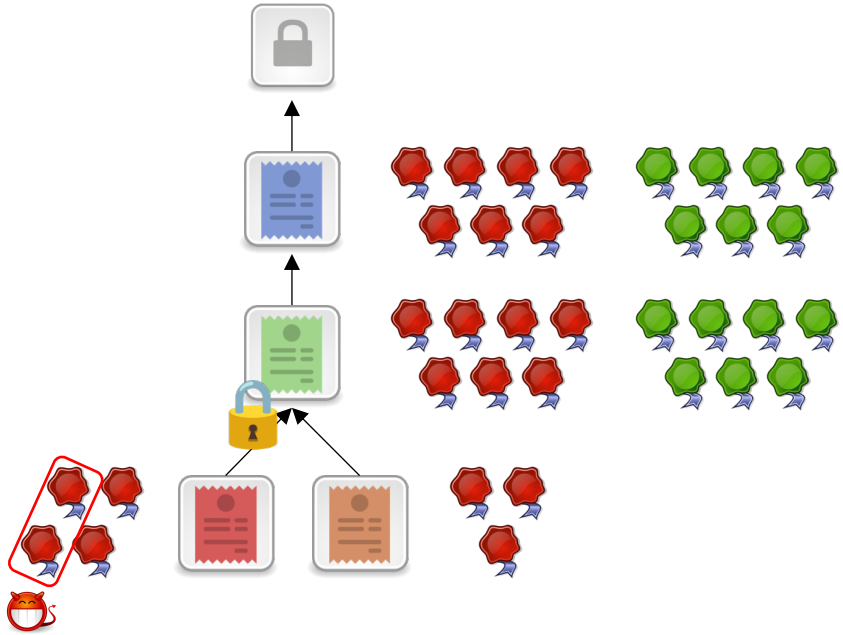


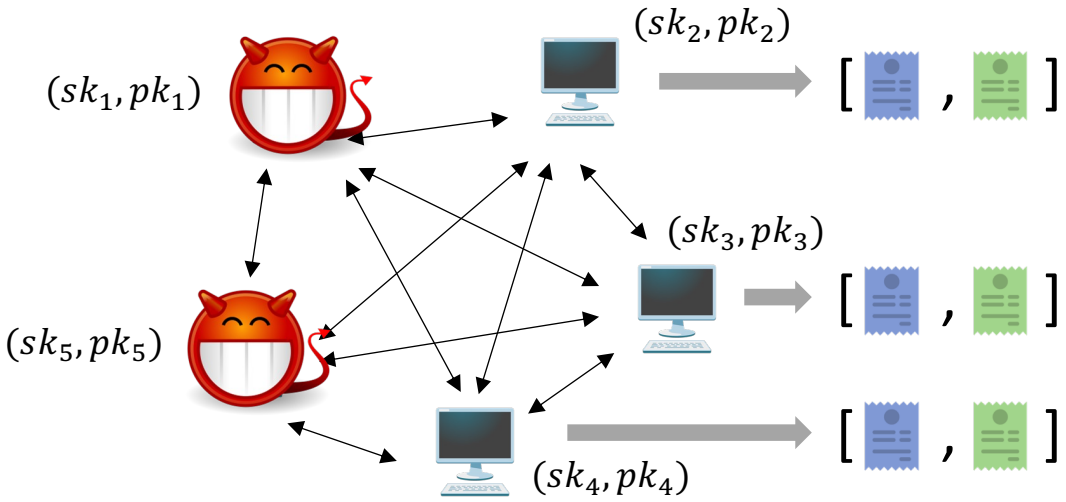


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

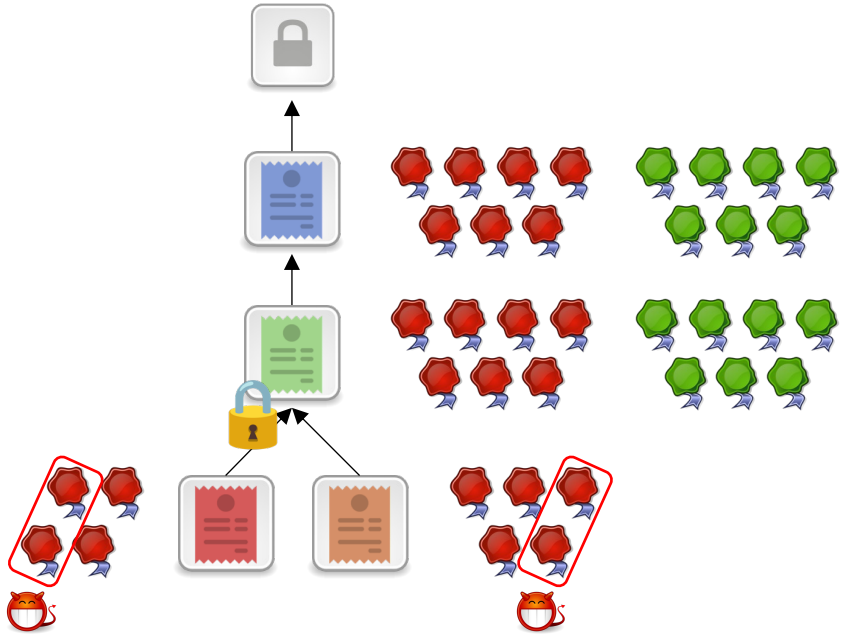


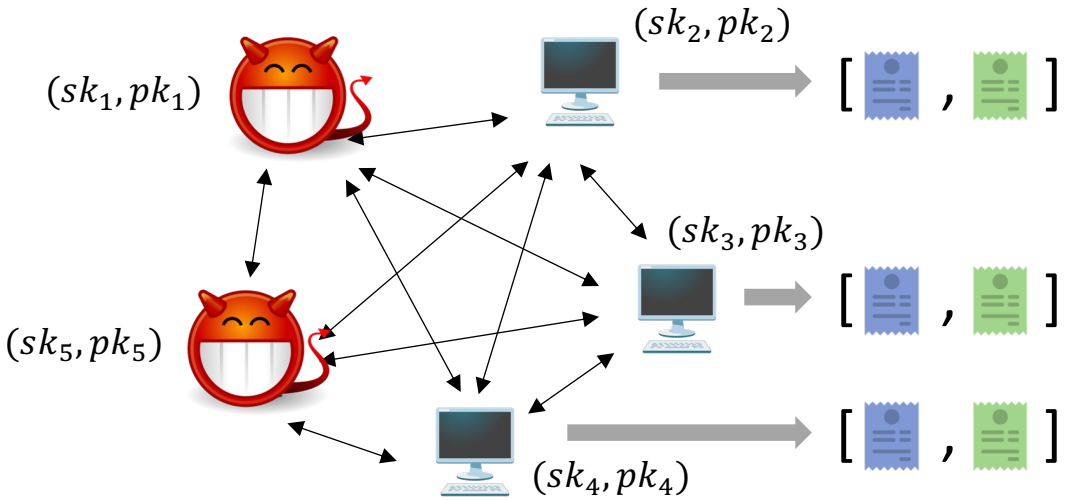


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

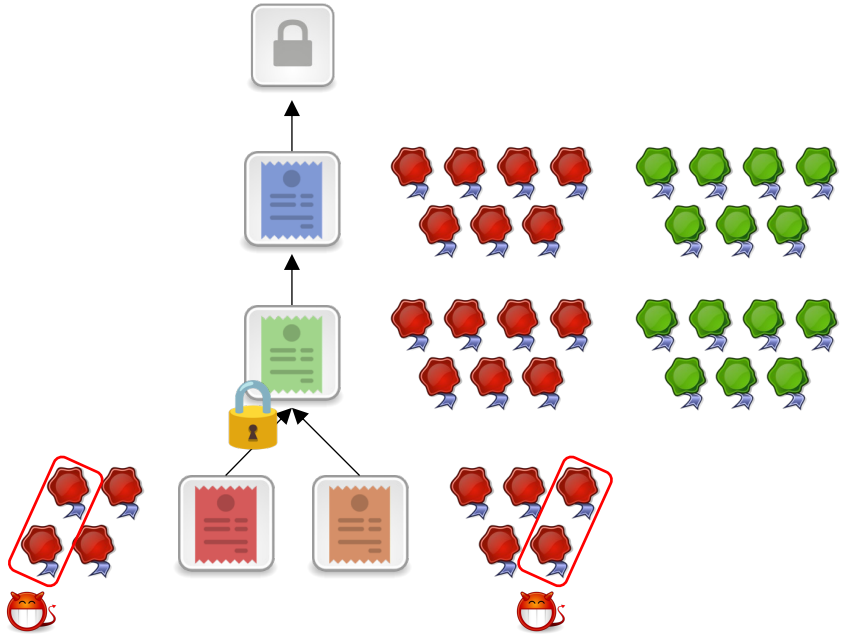


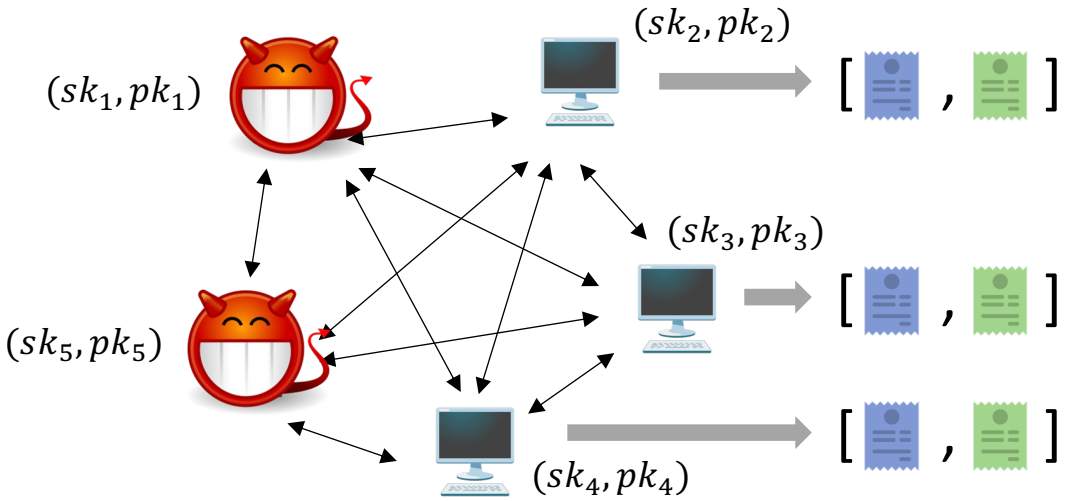


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

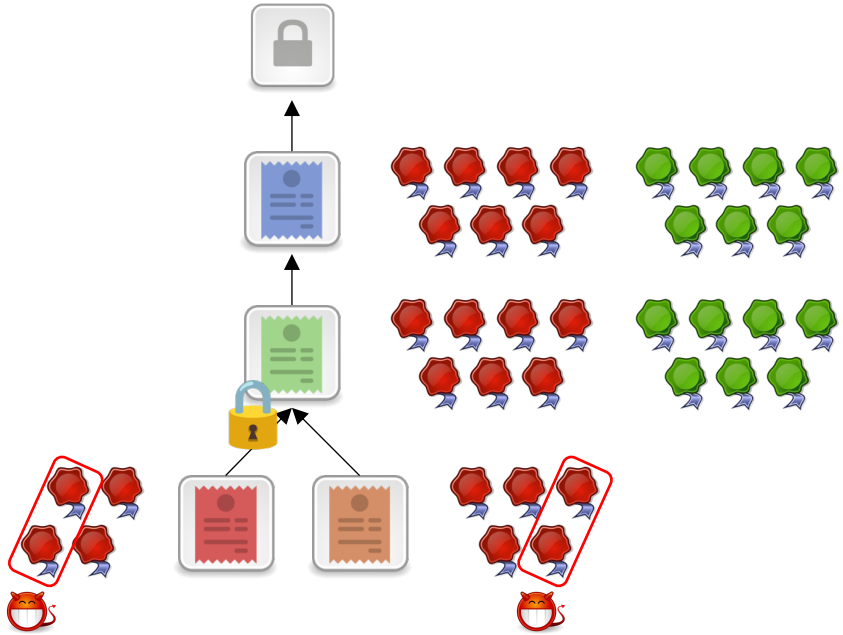


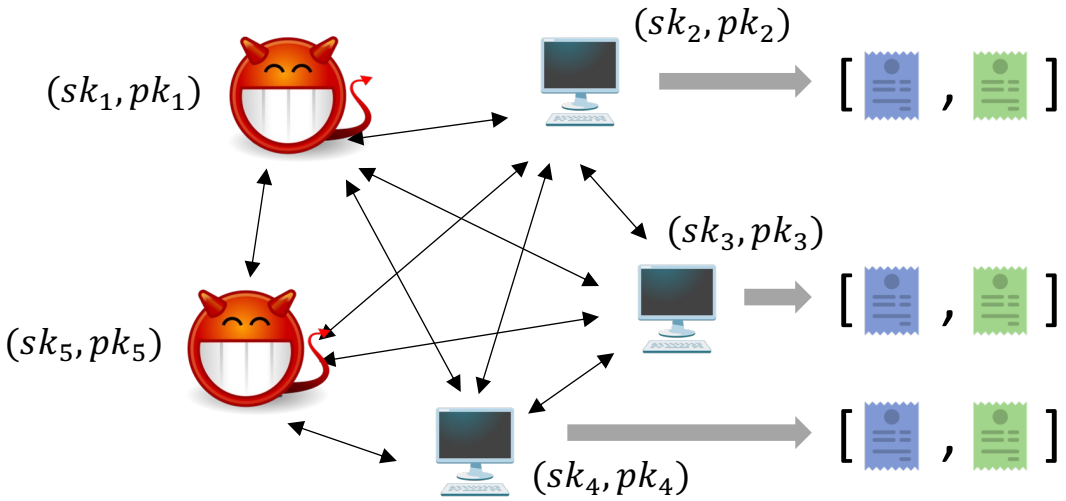


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

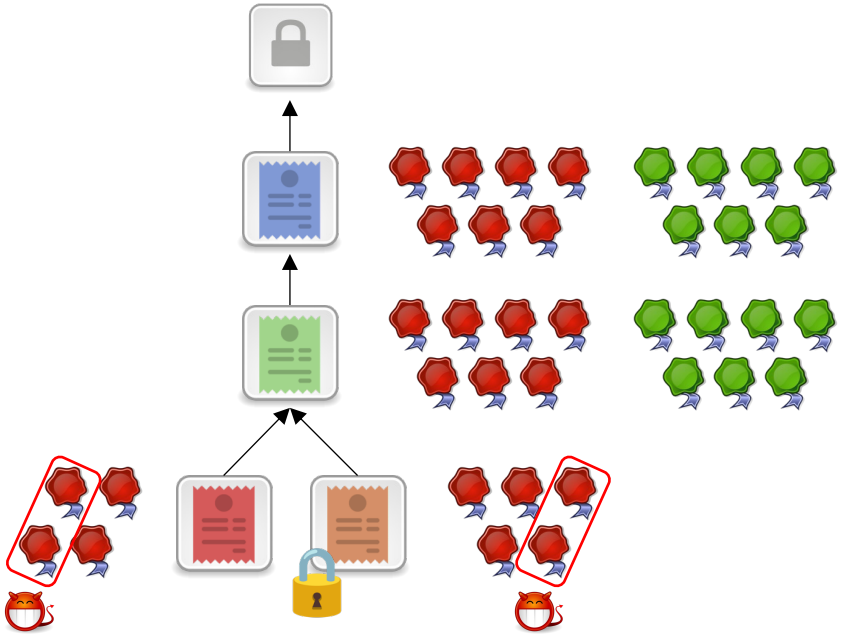


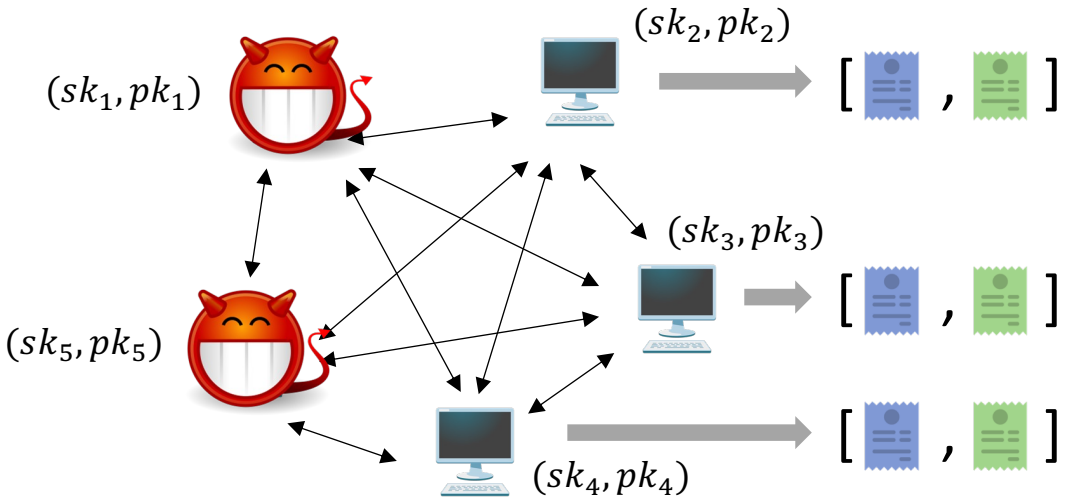


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

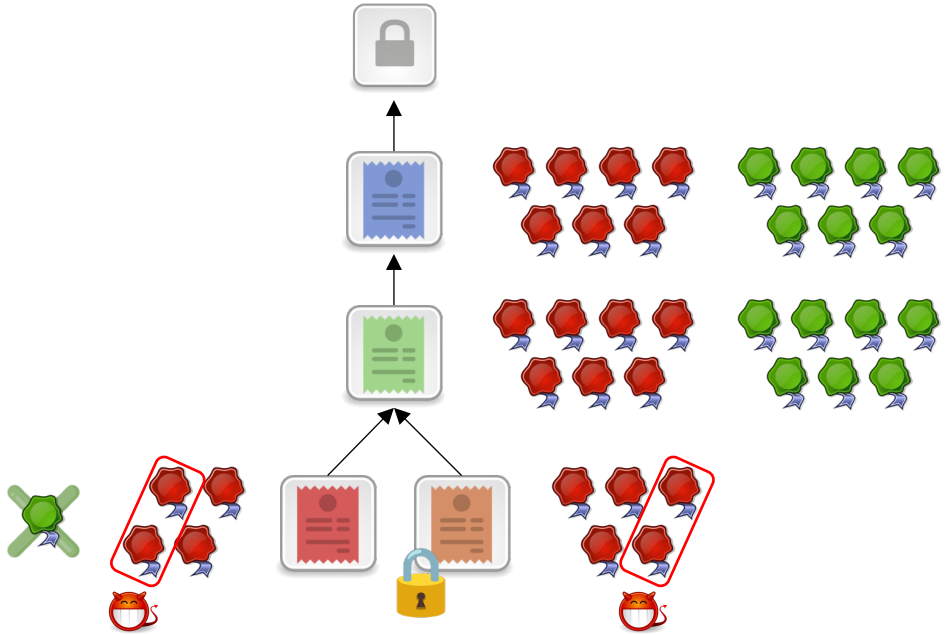


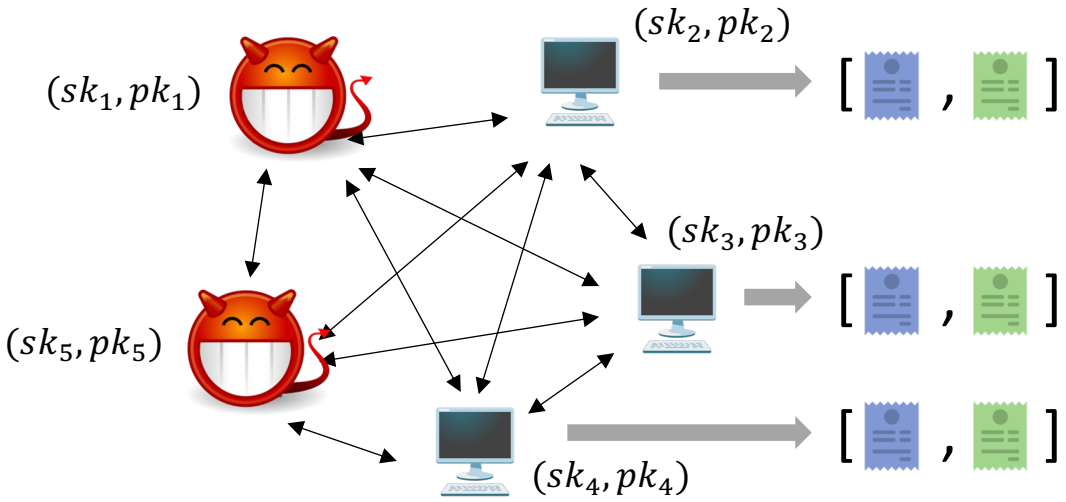


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

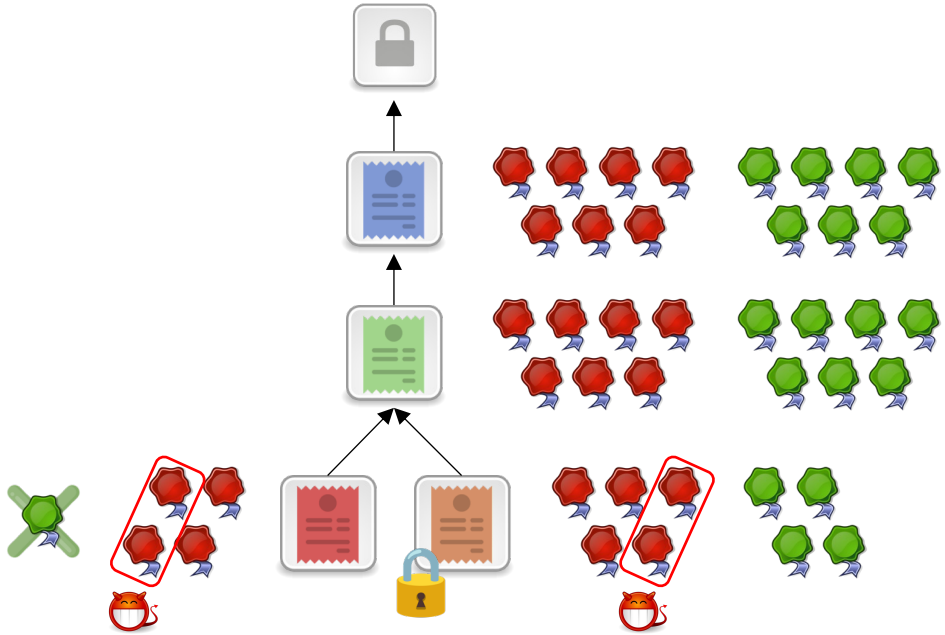


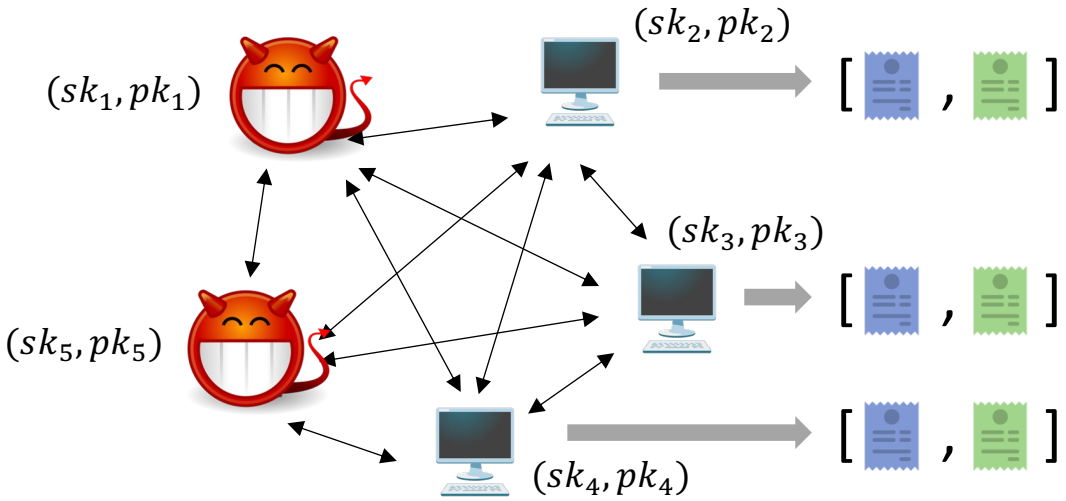


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

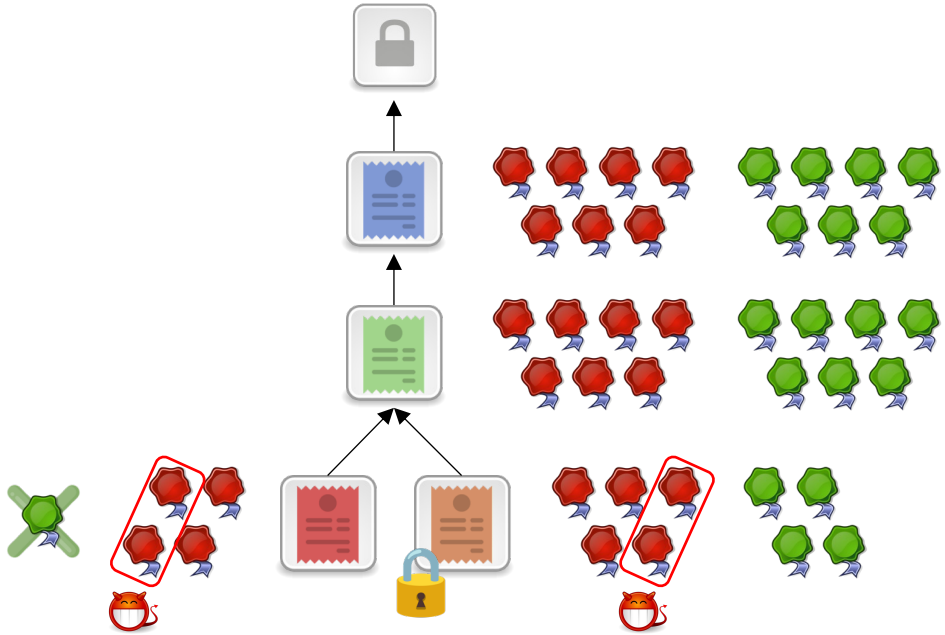


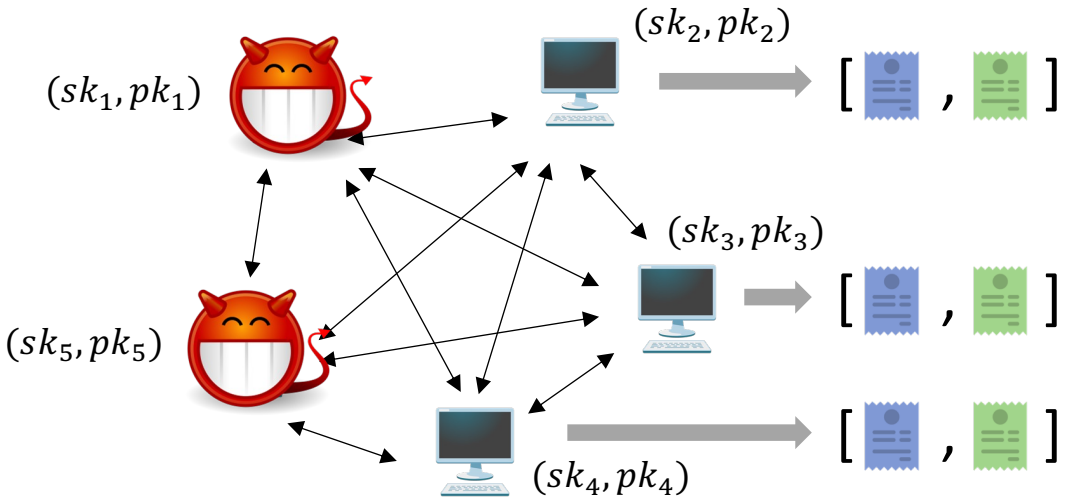


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

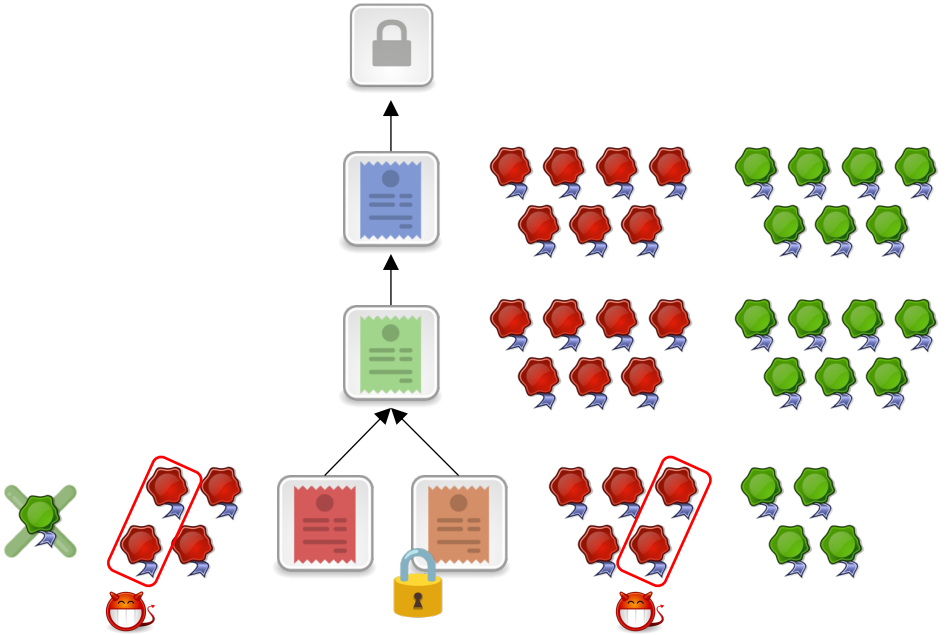


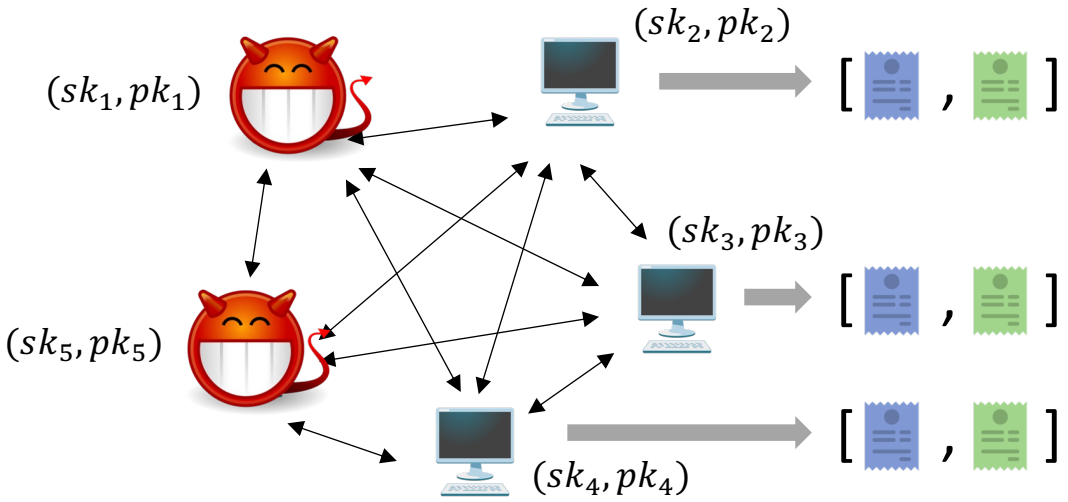


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

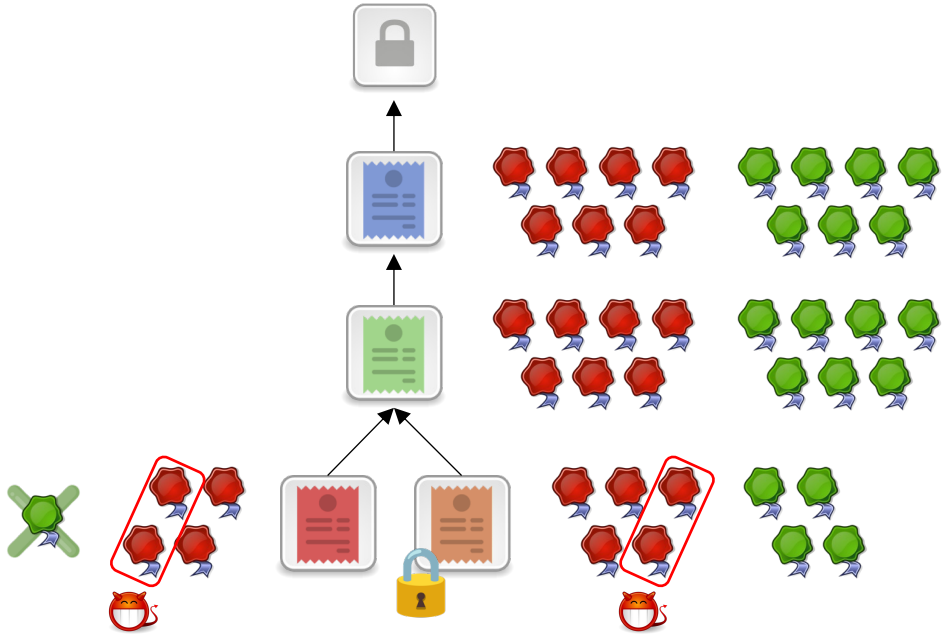


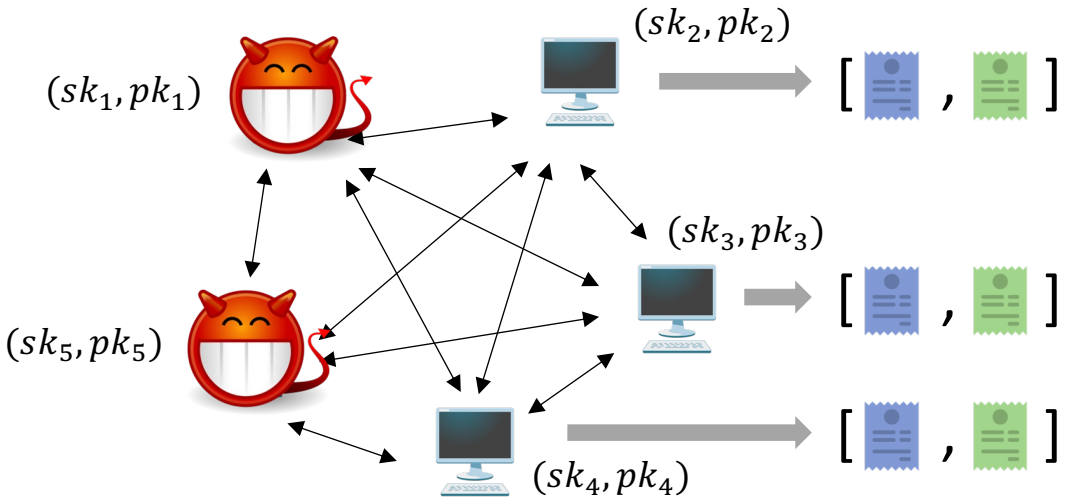


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

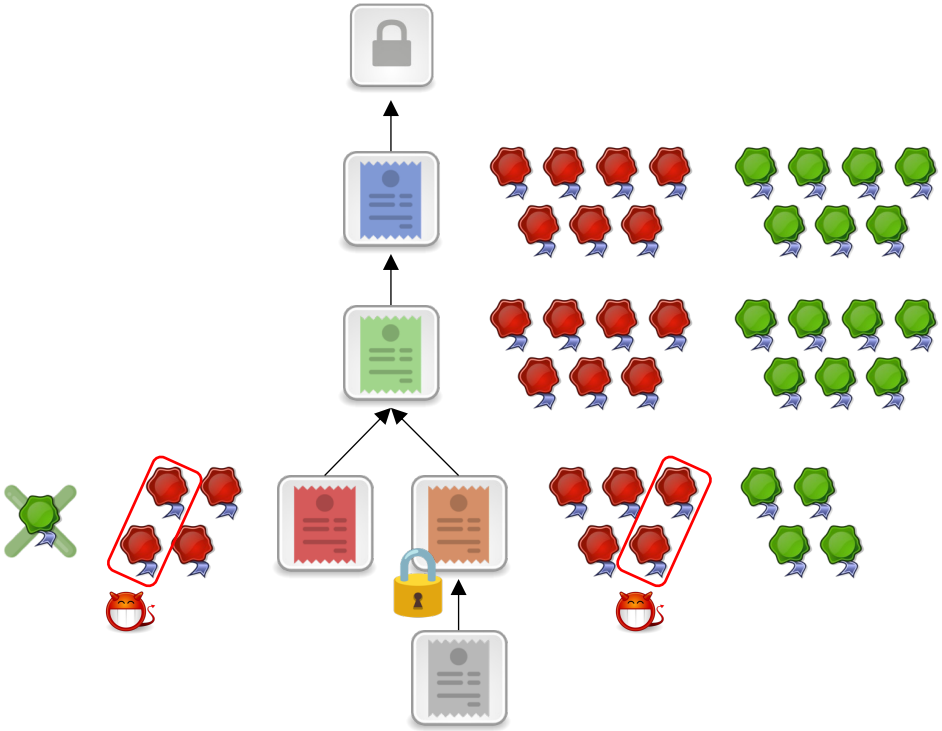


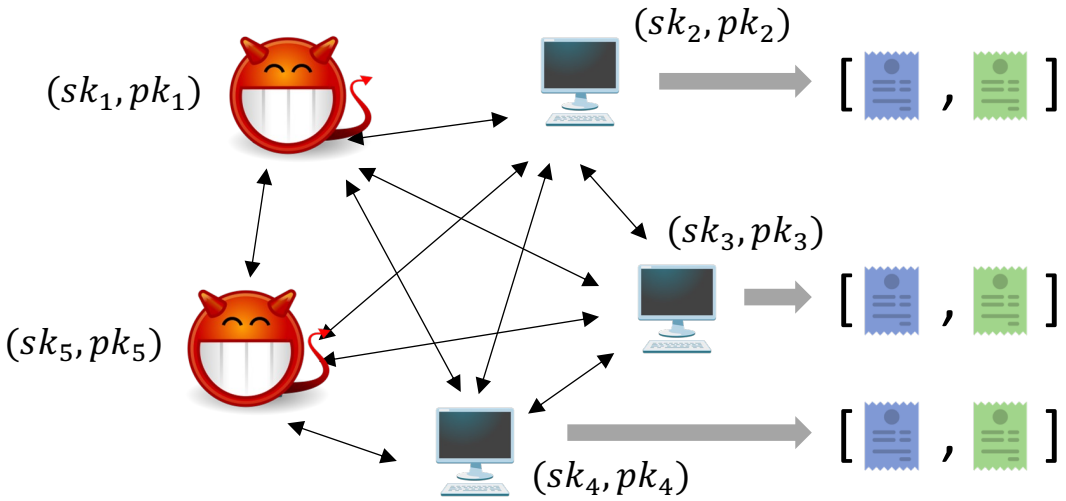


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

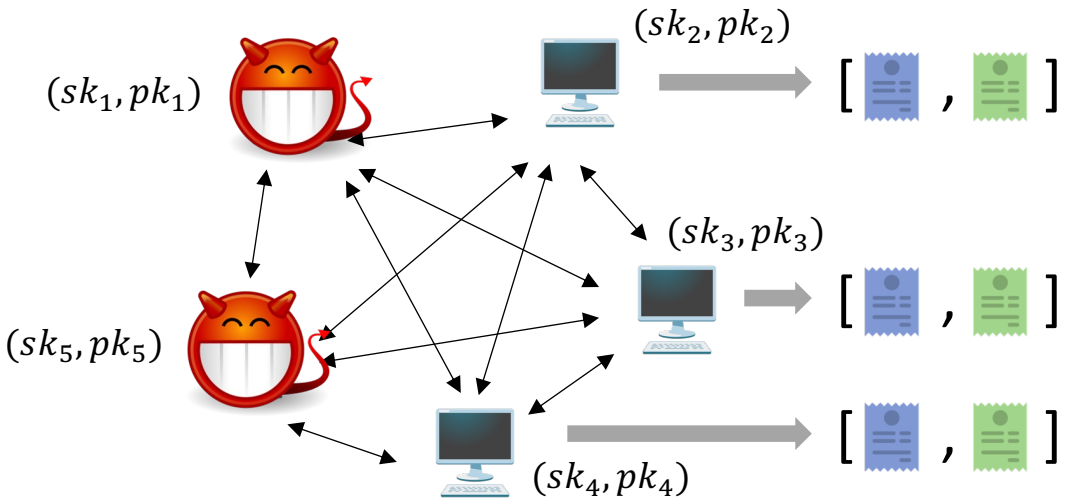


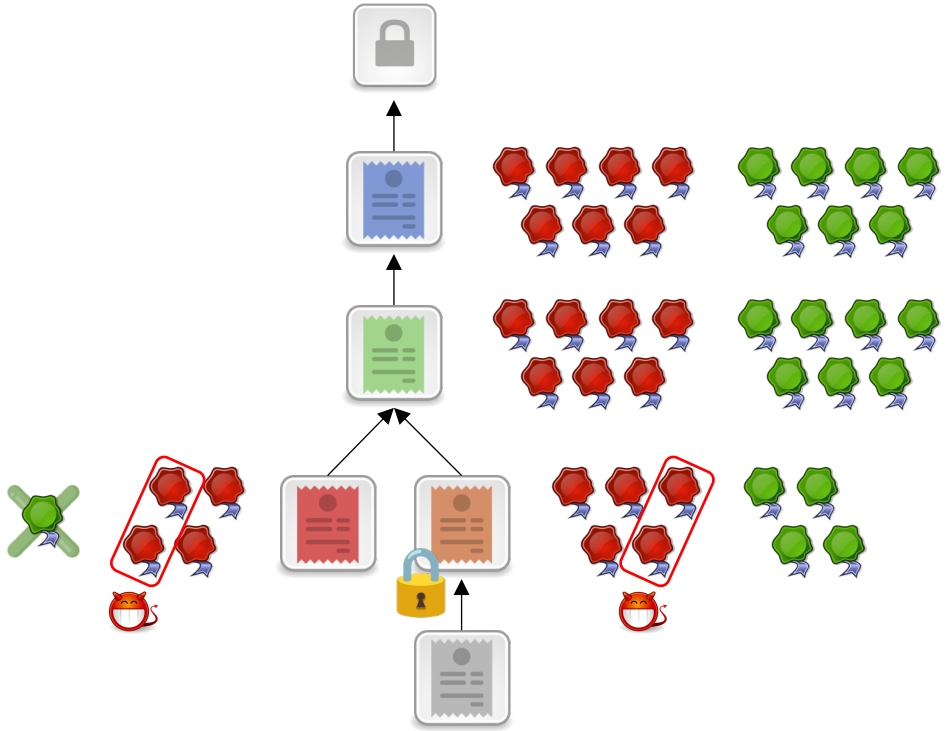


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

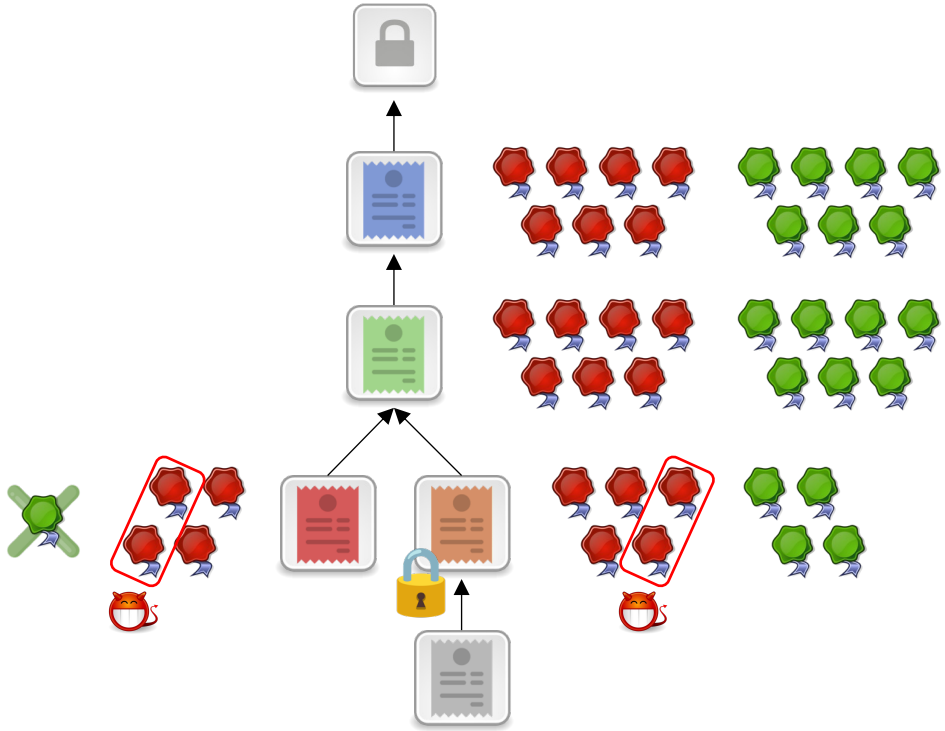


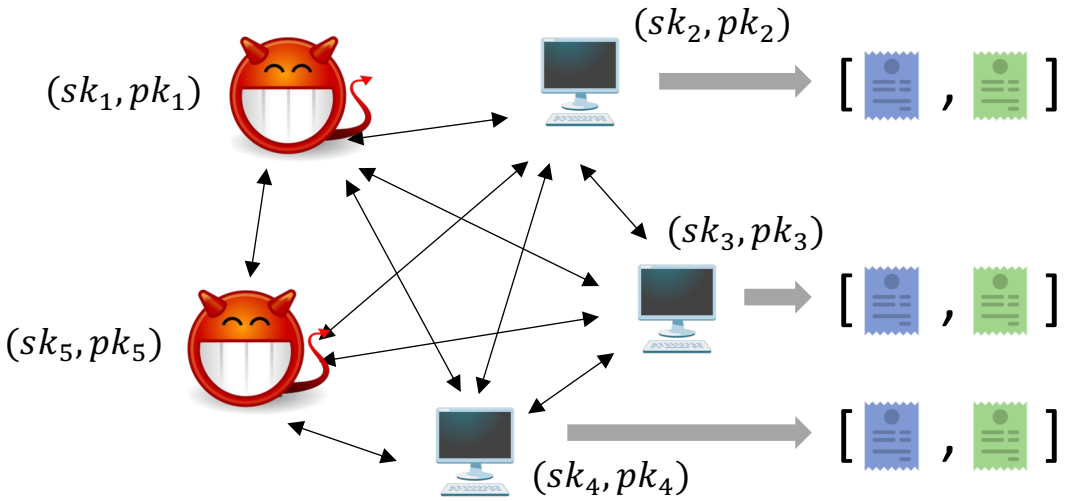


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

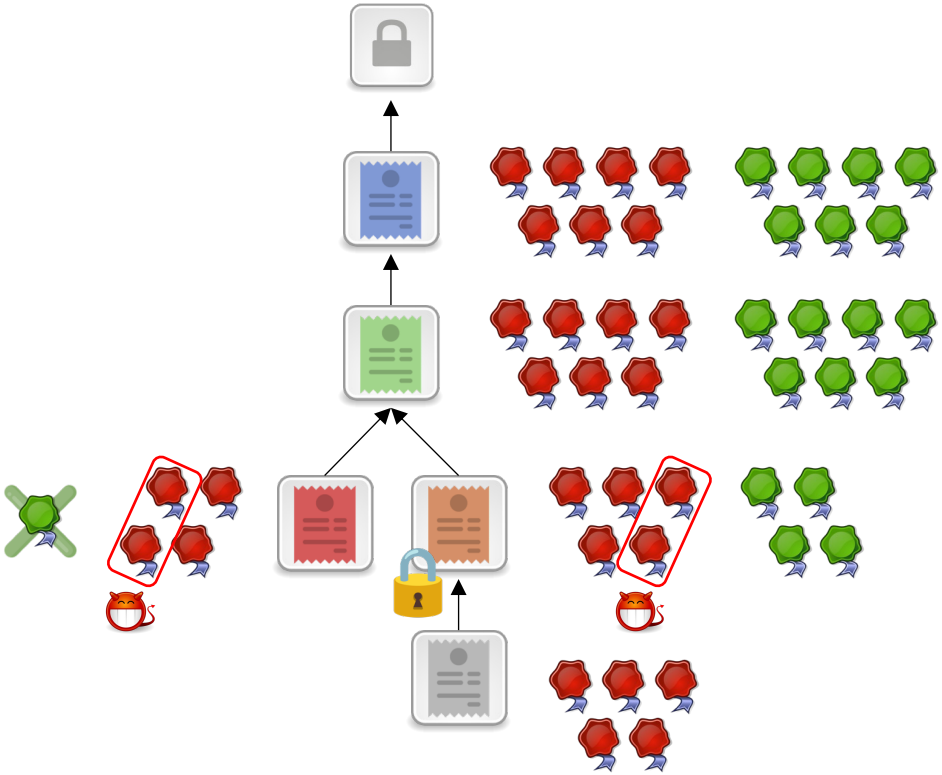


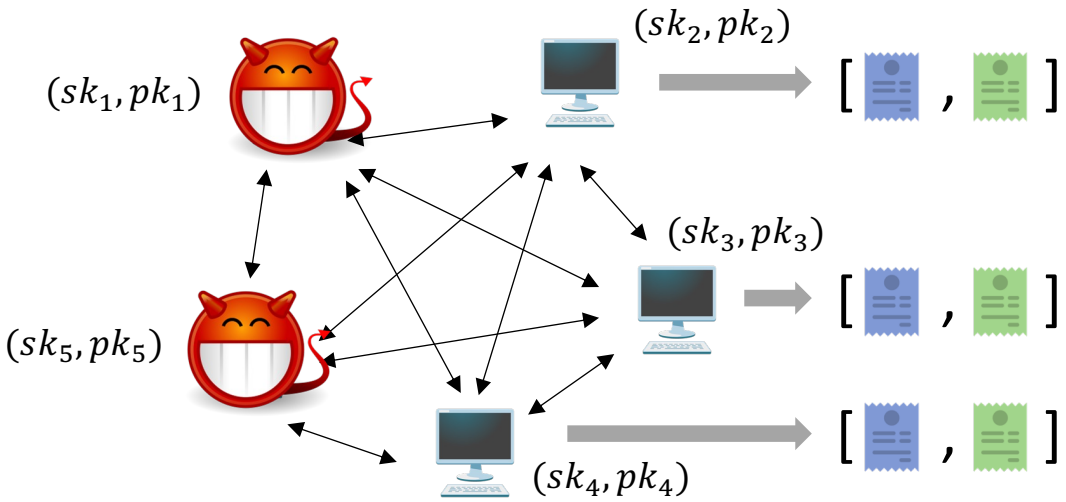


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

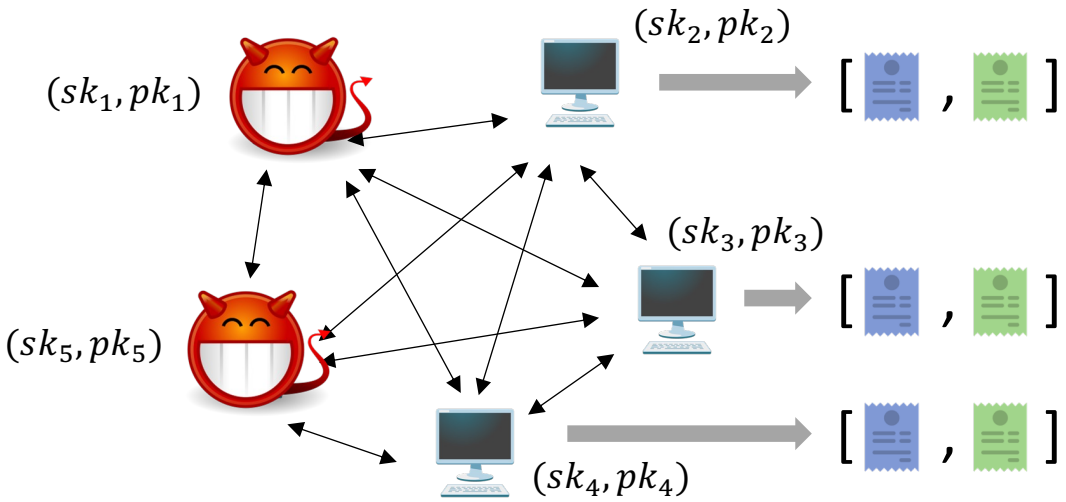


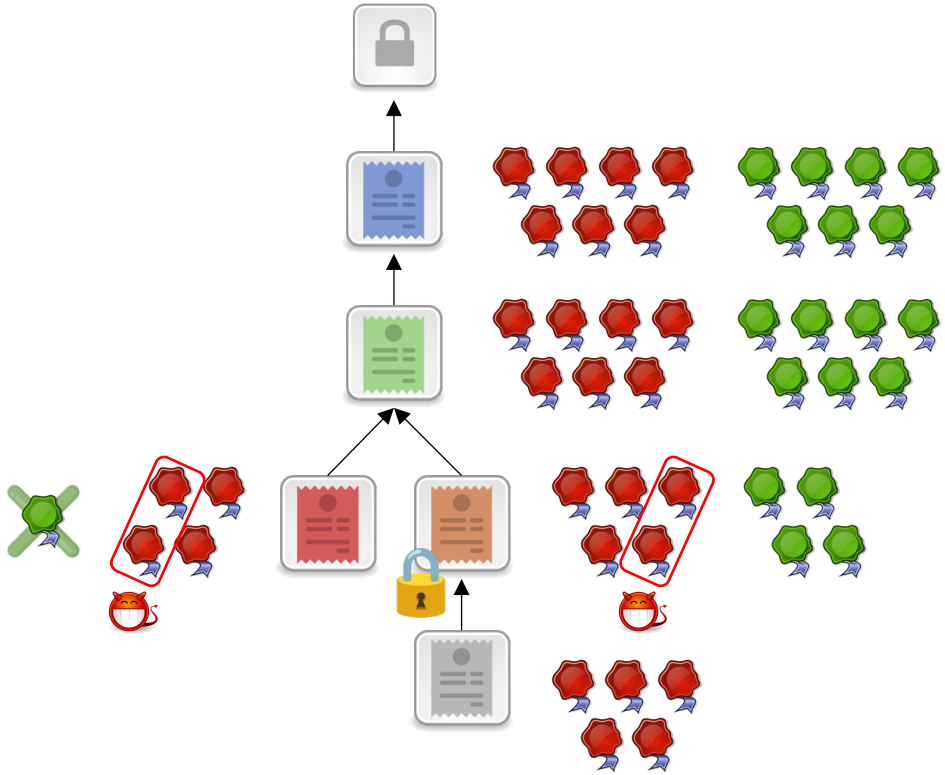


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

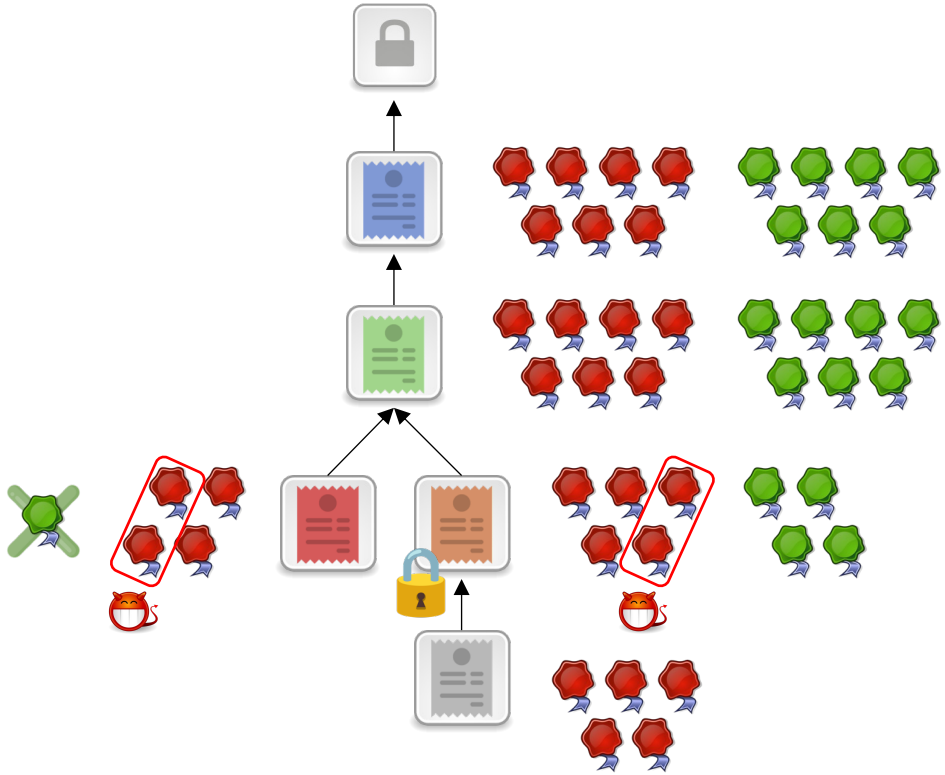


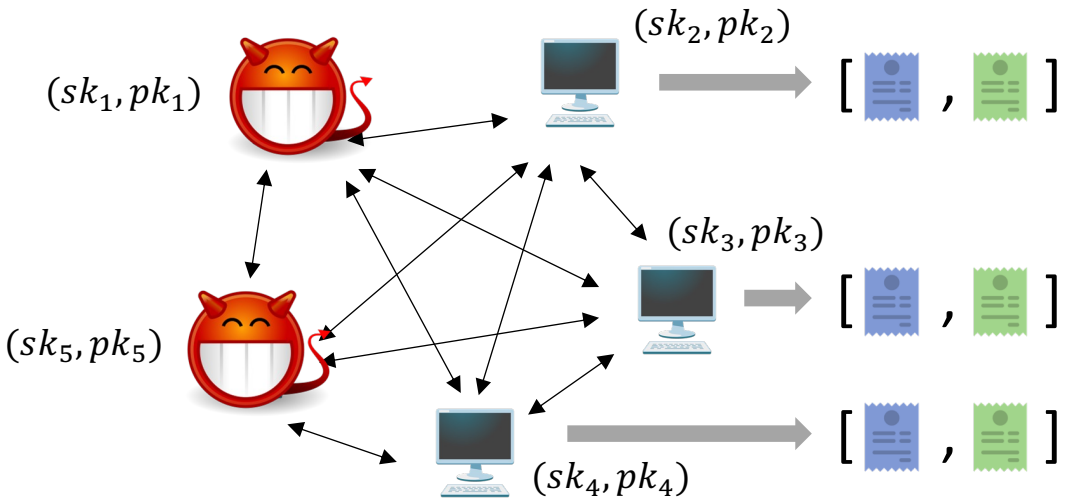


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

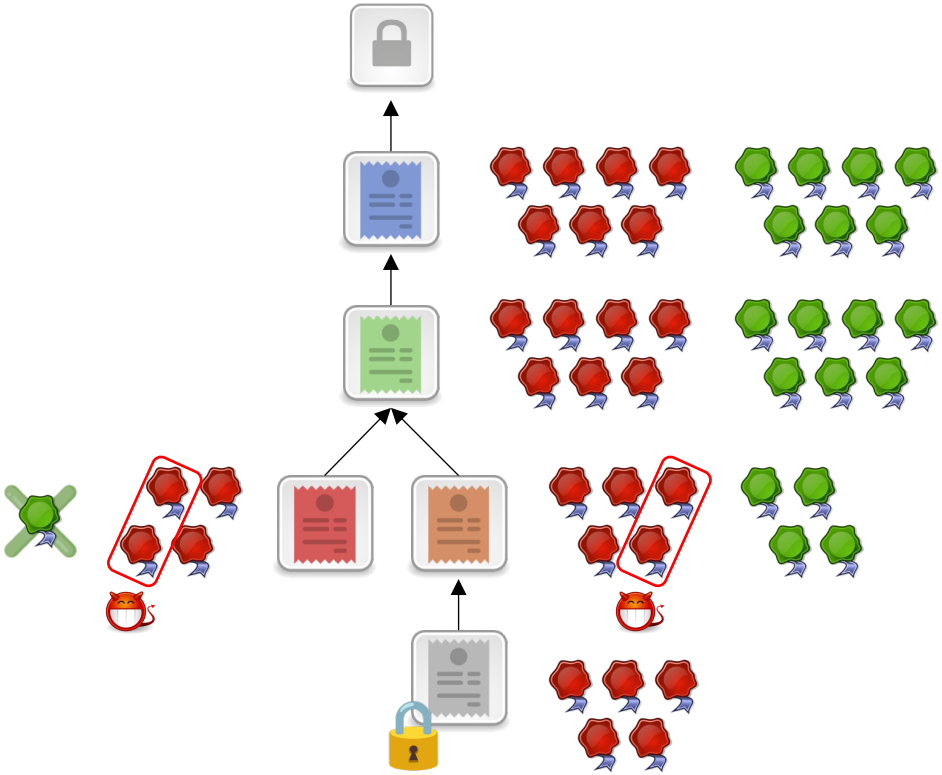


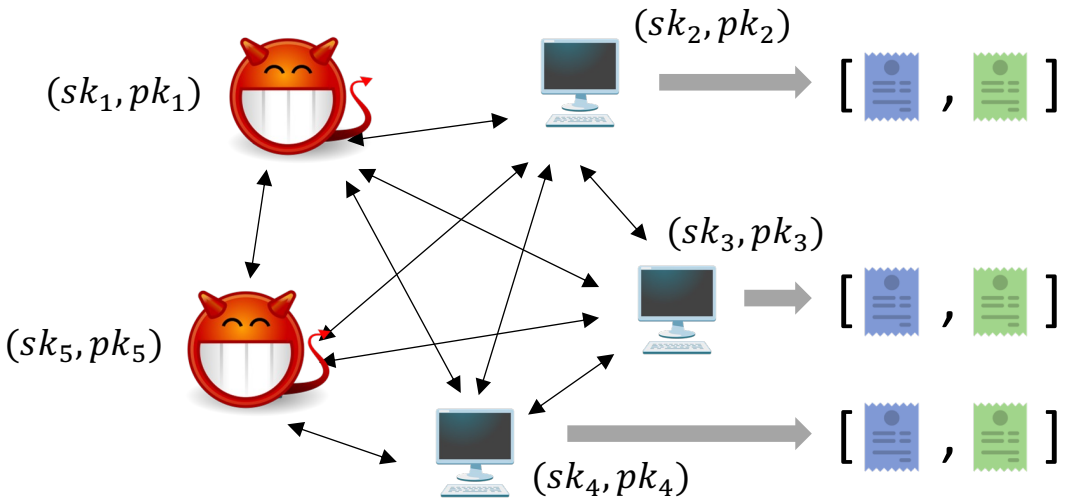


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

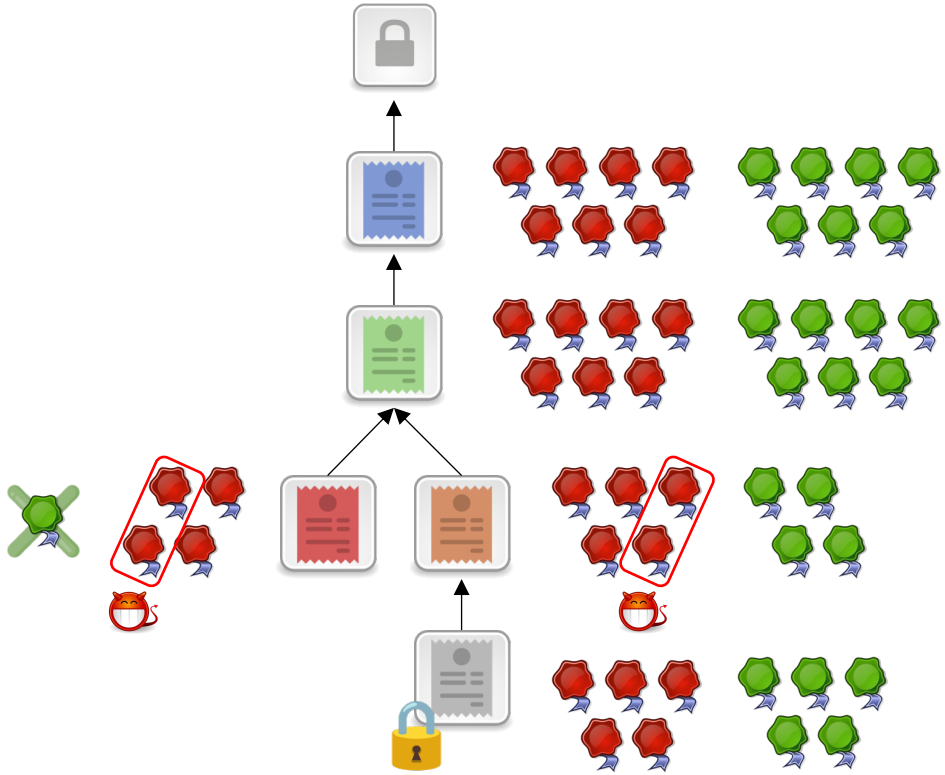


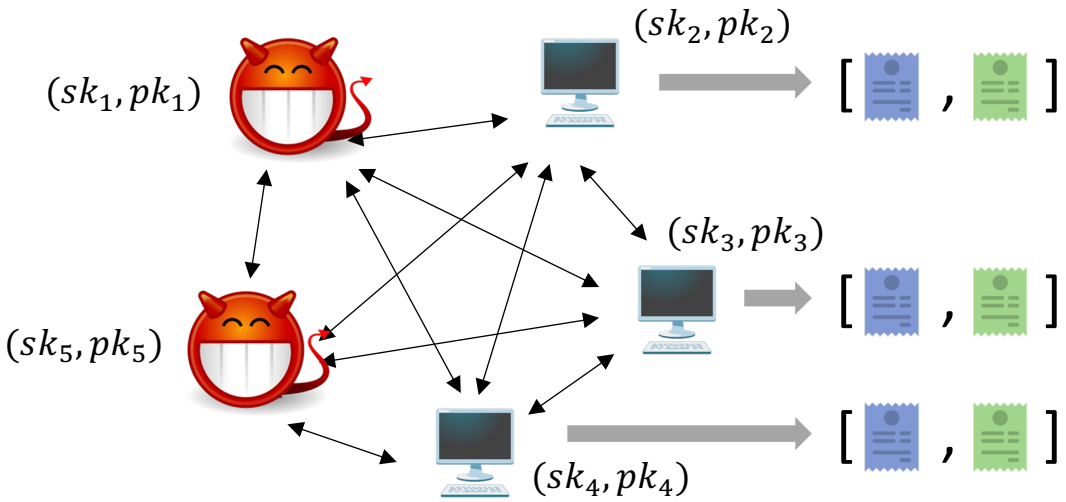


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

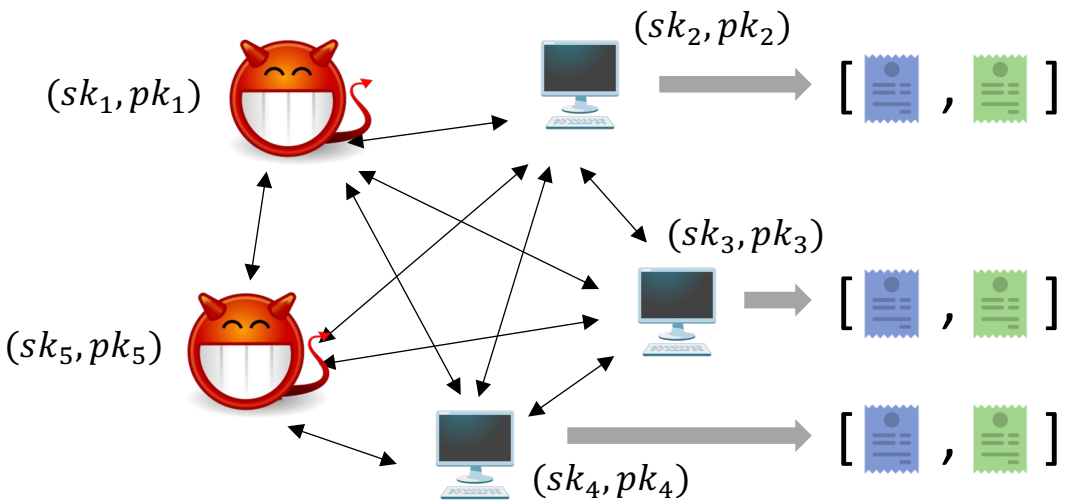


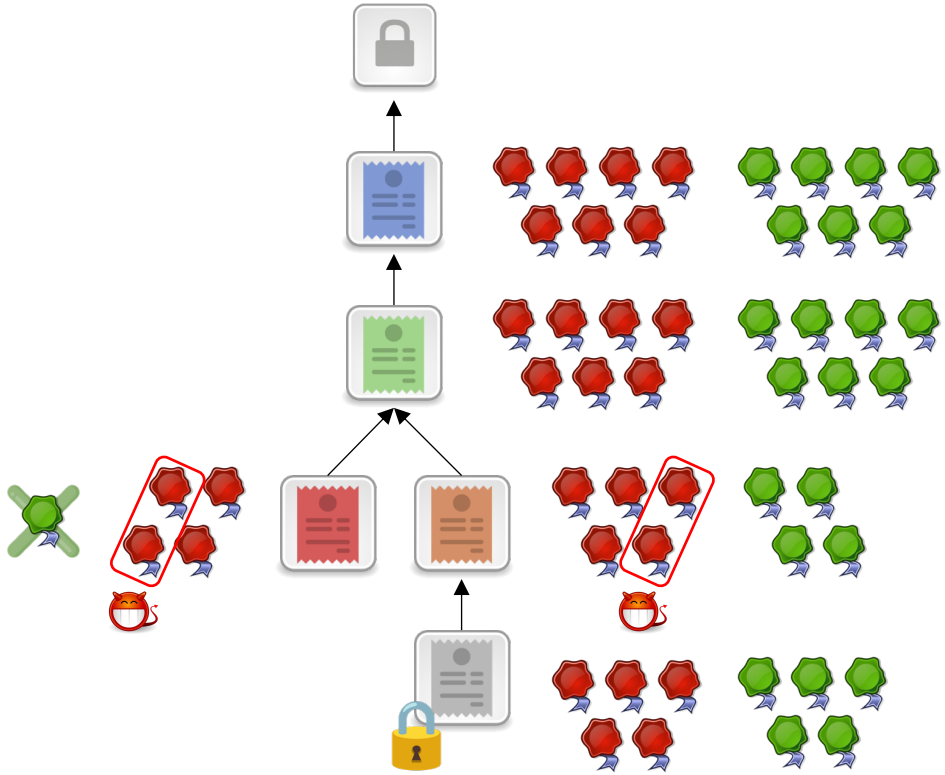


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

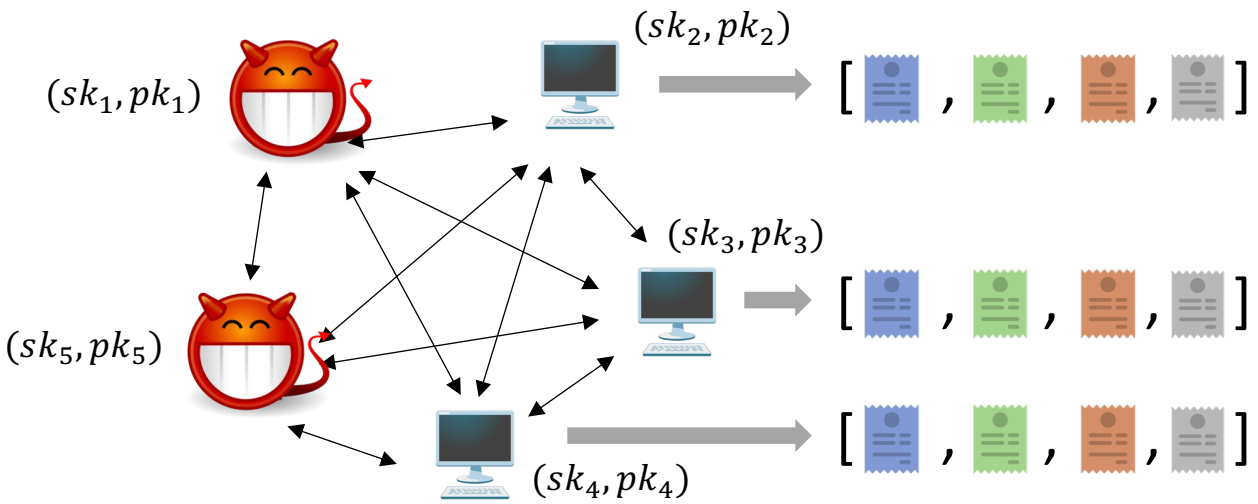


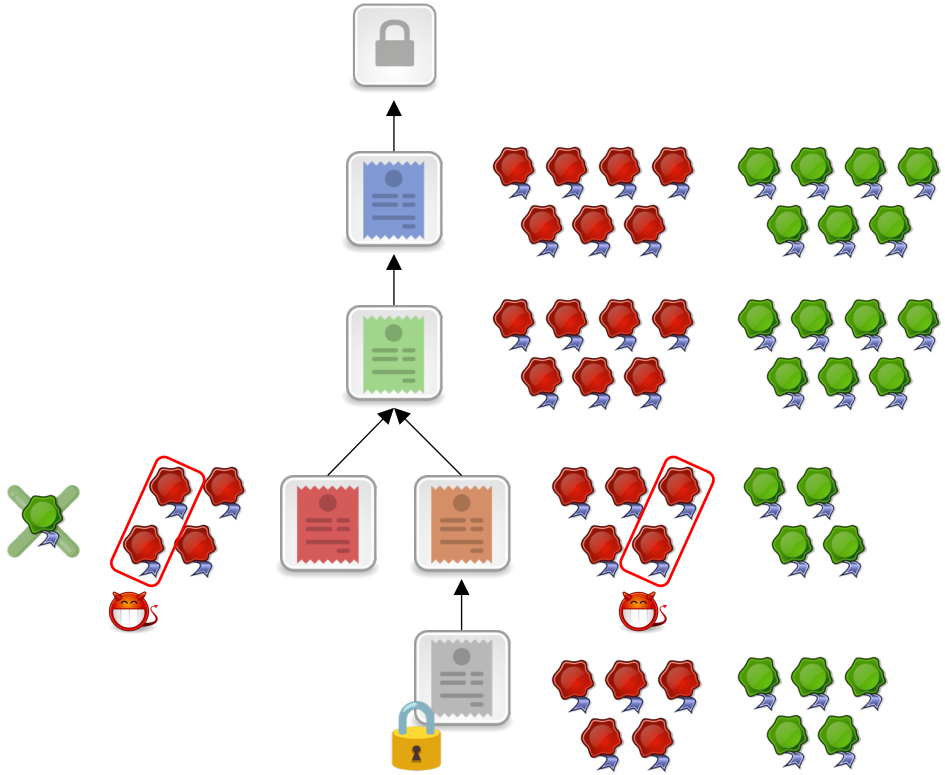


PBFT-Style Consensus

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}





Security

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

Security

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
    at time $t = 3\Delta v + 0\Delta$: // only leader of view v
        $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
        broadcast $NewBlock(v, b, Q, GetPendingTxs())$
    at time $t = 3\Delta v + 1\Delta$: // all nodes
        for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
            broadcast $NewVote(b, 1)$
    at time $t = 3\Delta v + 2\Delta$: // all nodes
        for vote-1 quorum Q for proposal b:
            $Q_{\text{lock}} \leftarrow Q$
            broadcast $NewVote(b, 2)$, broadcast Q_{lock}

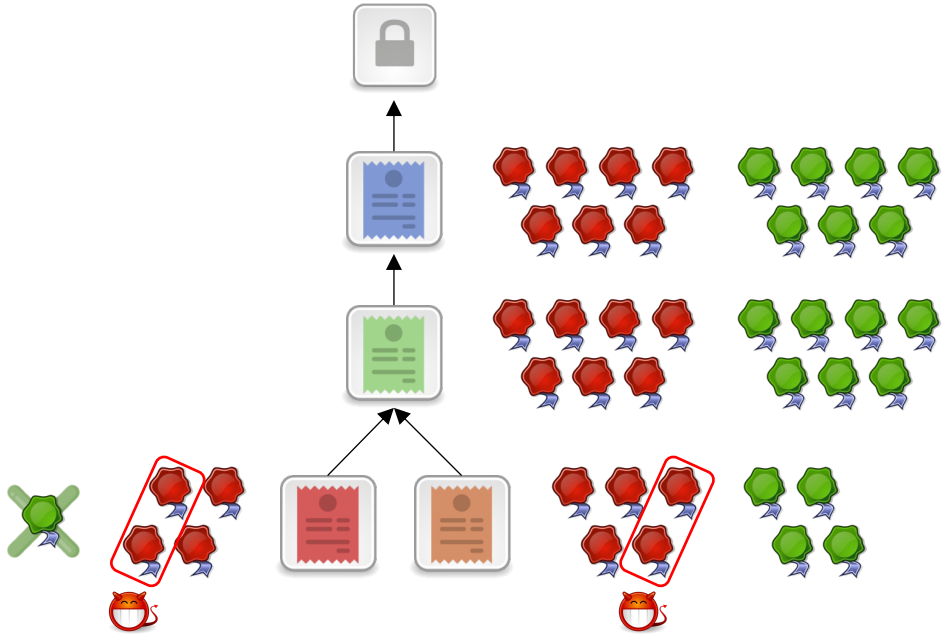
Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

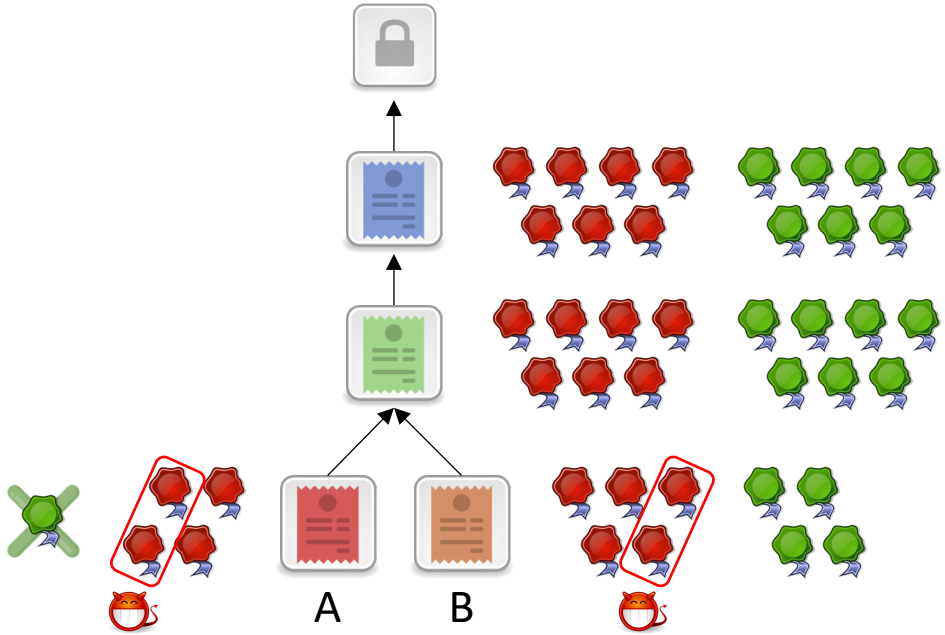
- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

Safety:

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

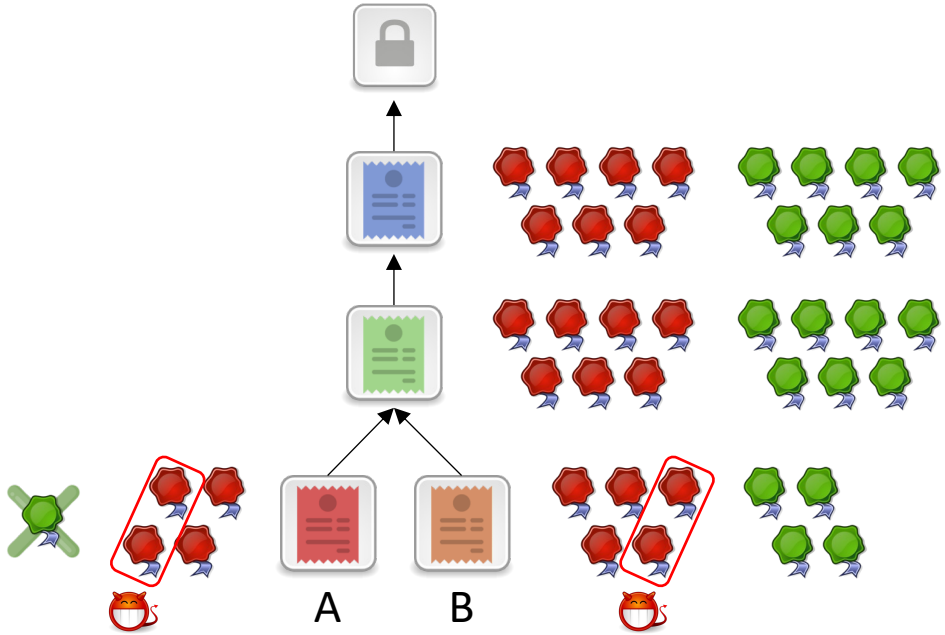
- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

Safety:

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

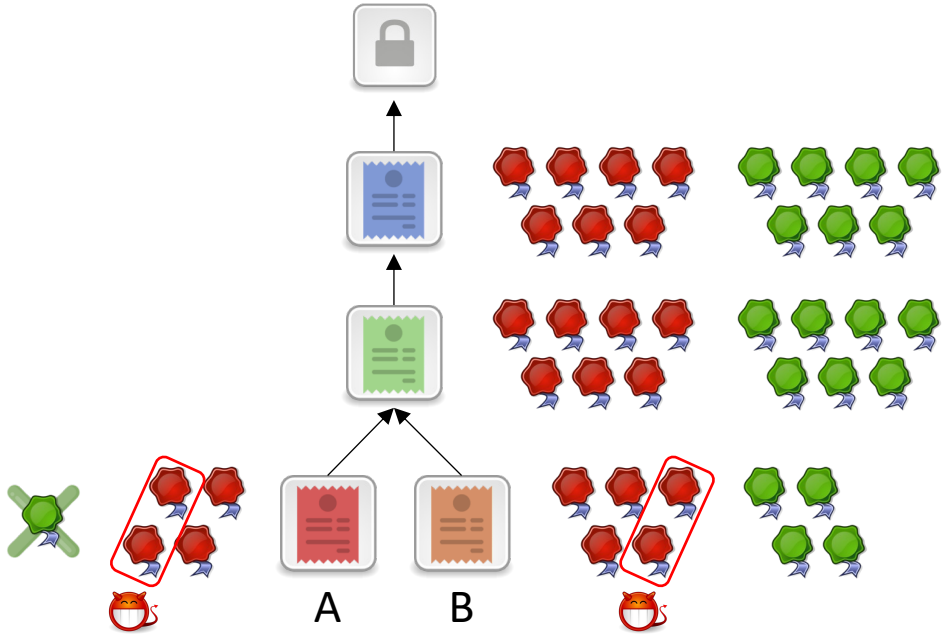
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

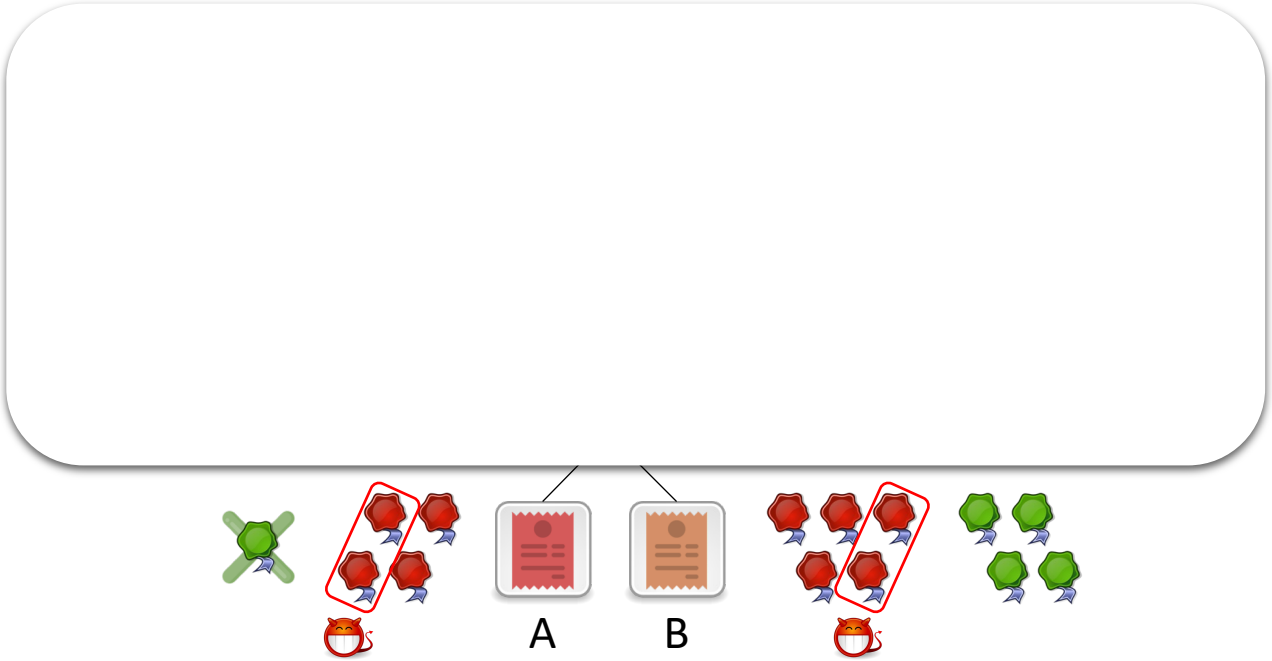
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

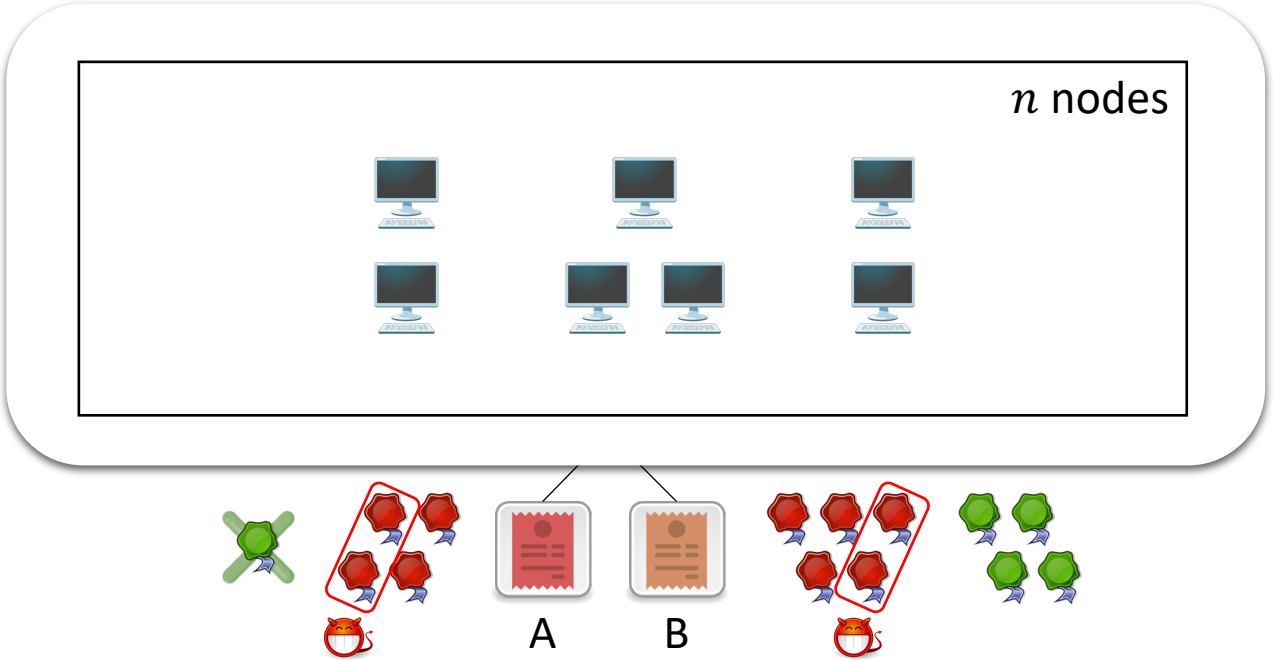
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

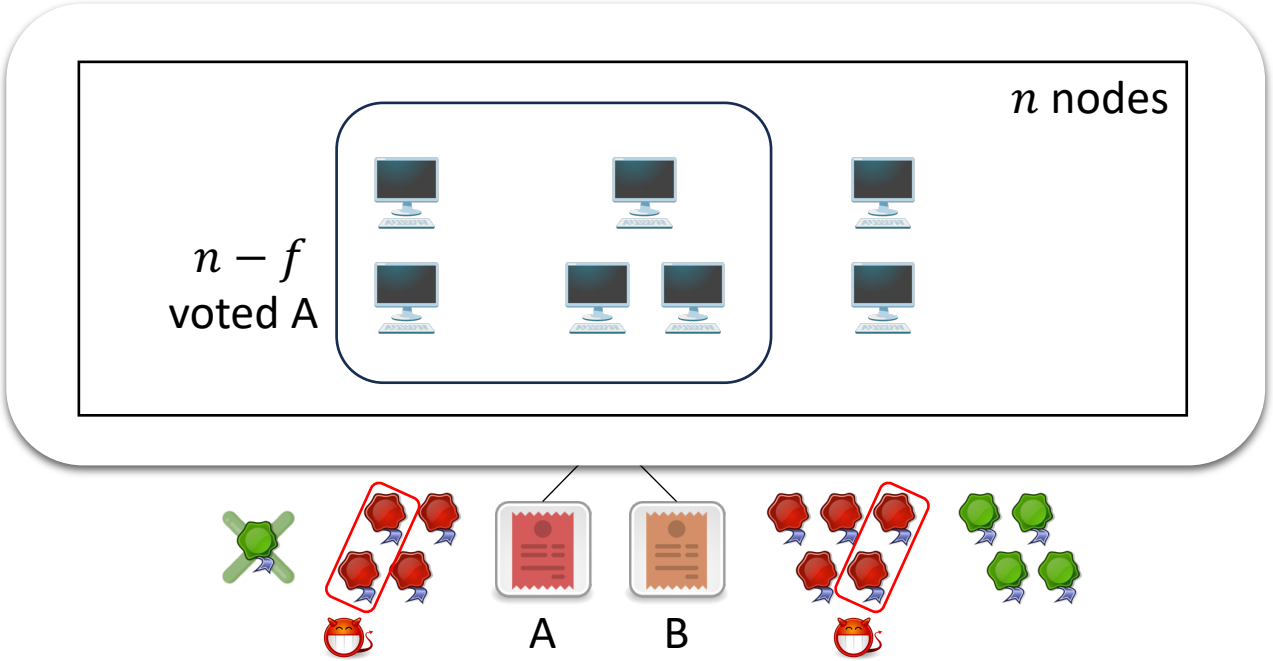
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

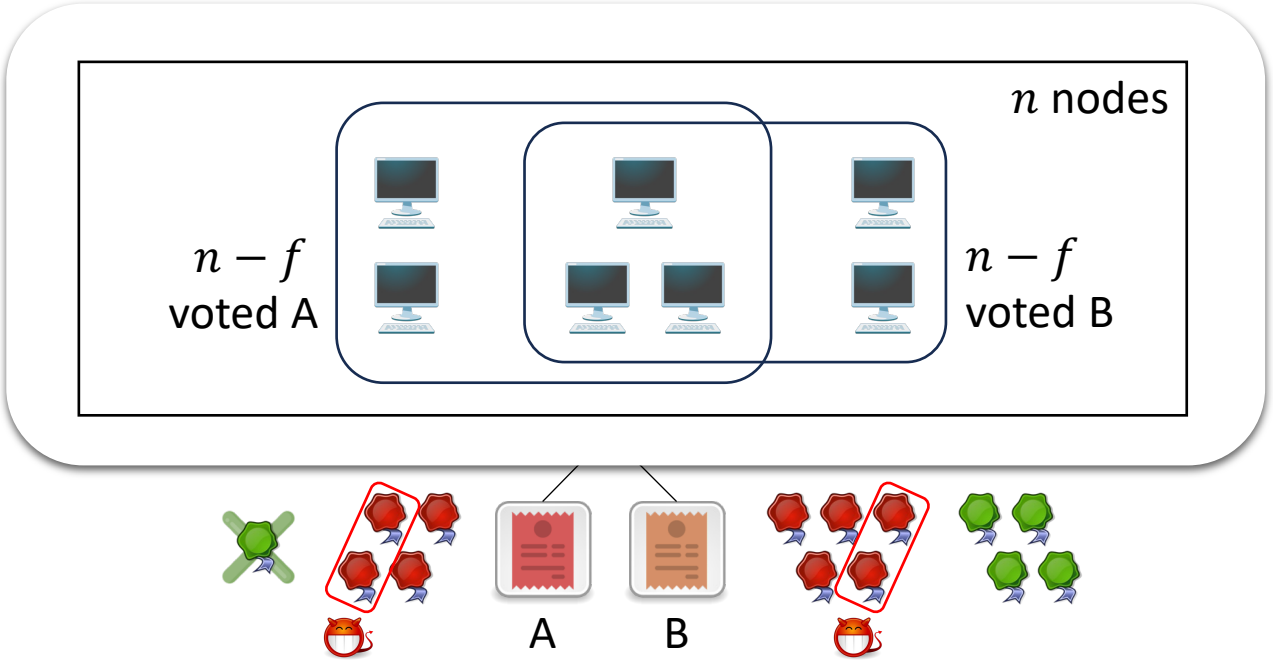
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …
- → At least $n - 2f > f$ nodes 1-voted both A and B.

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

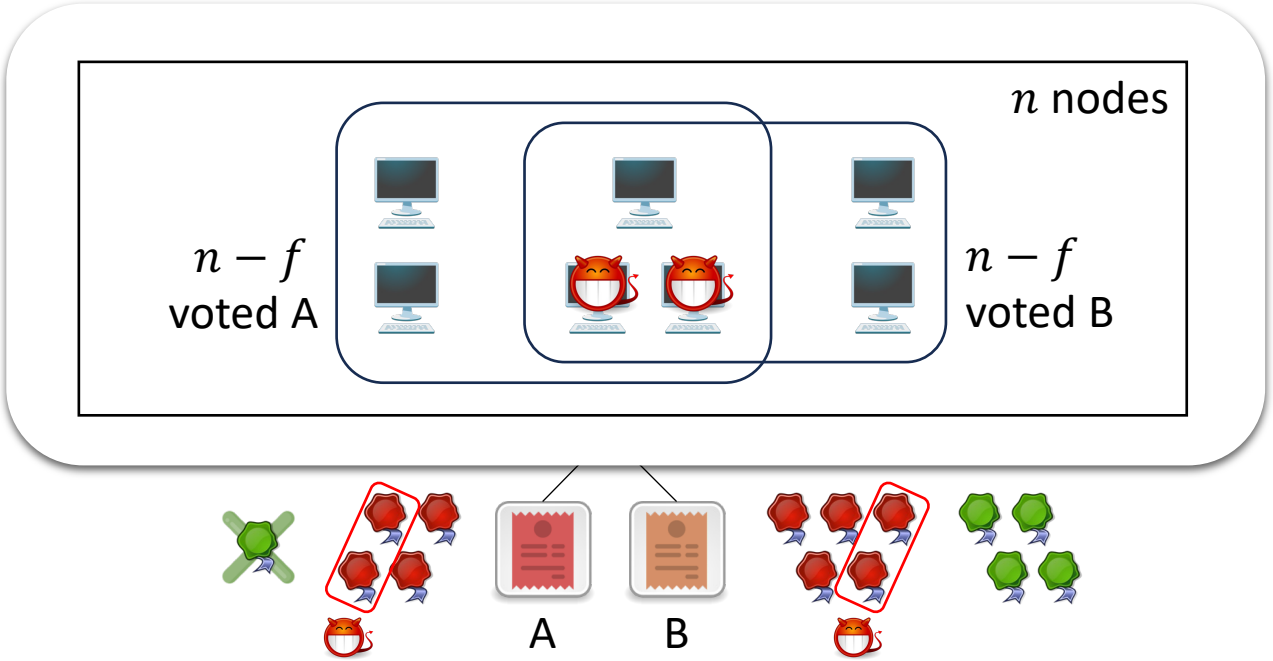
Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …
- → At least $n - 2f > f$ nodes 1-voted both A and B.

Security

Example: $n = 7, f = 2$ → quorum 5

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

Safety:

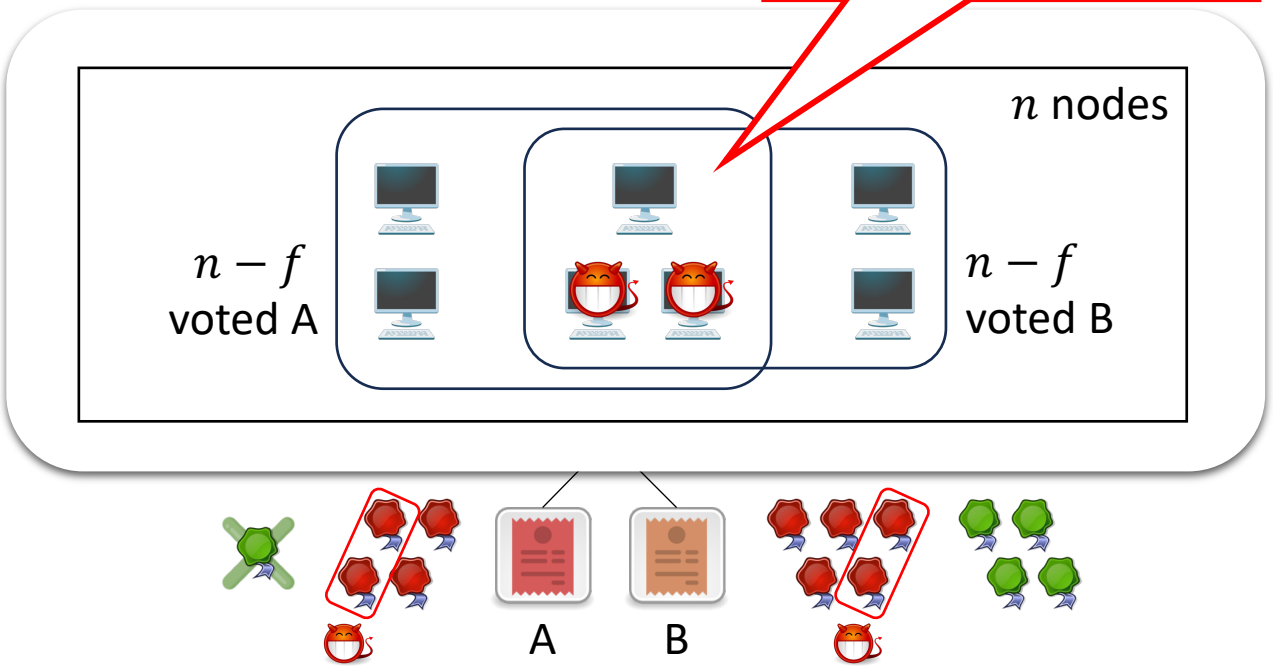
- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …
- → At least $n - 2f > f$ nodes 1-voted both A and B.
- → An honest node 1-voted both A and B.
- Contradiction! → No conflicting blocks confirmed!

Security

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

Example: $n = 7, f = 2 \rightarrow$ quorum 5

"Quorum intersection argument"



Liveness:

- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Proposal b satisfies $view(Q_{\text{lock}}) \leq view(b.Q)$ for all honest nodes. → All honest nodes vote-1 for b.
- All honest nodes see a vote-1 quorum for b.
  → All honest nodes vote-2 for b.
- → Honest proposal b gets confirmed.

Safety:

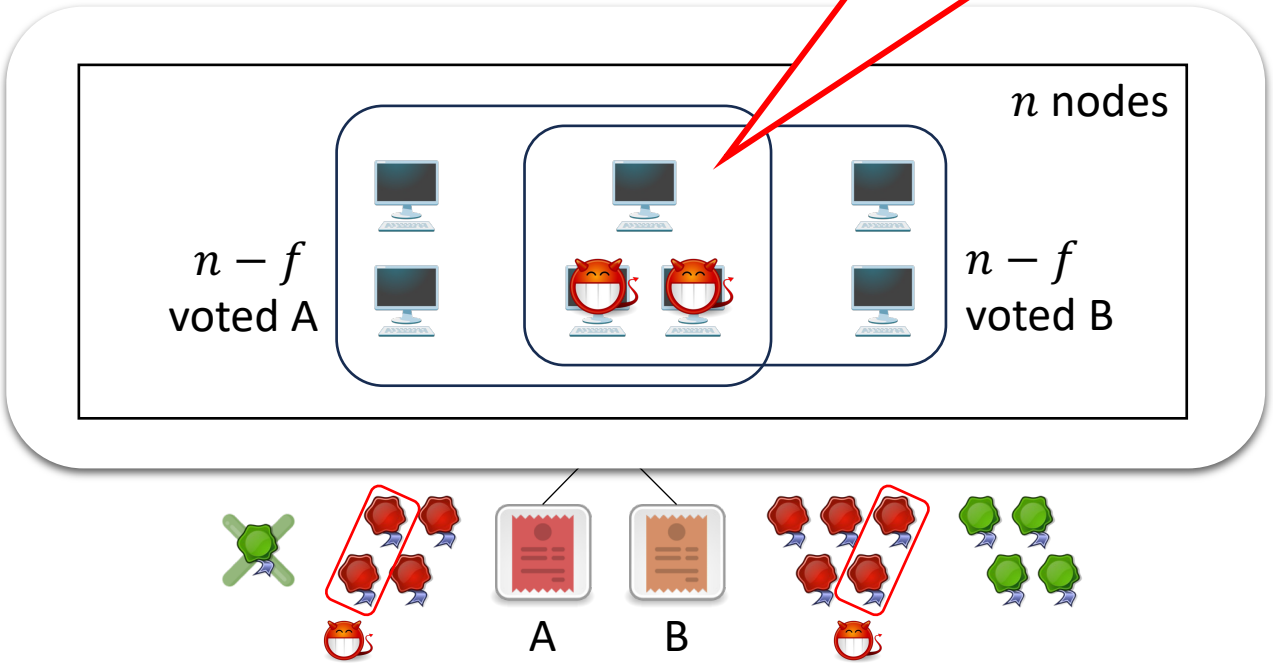
- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …
- → At least $n - 2f > f$ nodes 1-voted both A and B.
- → An honest node 1-voted both A and B.
- Contradiction! → No conflicting blocks confirmed!

Security

$Q_{\text{lock}} \leftarrow \{\}$
for view $v = 1,2,3, \ldots$:
  at time $t = 3\Delta v + 0\Delta$: // only leader of view v
    $(b, Q) \leftarrow$ block with vote-1 quorum for highest view
    broadcast $NewBlock(v, b, Q, GetPendingTxs())$
  at time $t = 3\Delta v + 1\Delta$: // all nodes
    for first proposal b with $view(Q_{\text{lock}}) \leq view(b.Q)$:
      broadcast $NewVote(b, 1)$
  at time $t = 3\Delta v + 2\Delta$: // all nodes
    for vote-1 quorum Q for proposal b:
      $Q_{\text{lock}} \leftarrow Q$
      broadcast $NewVote(b, 2)$, broadcast Q_{lock}

Example: $n = 7, f = 2 \rightarrow$ quorum 5

"Quorum intersection argument"



Liveness:

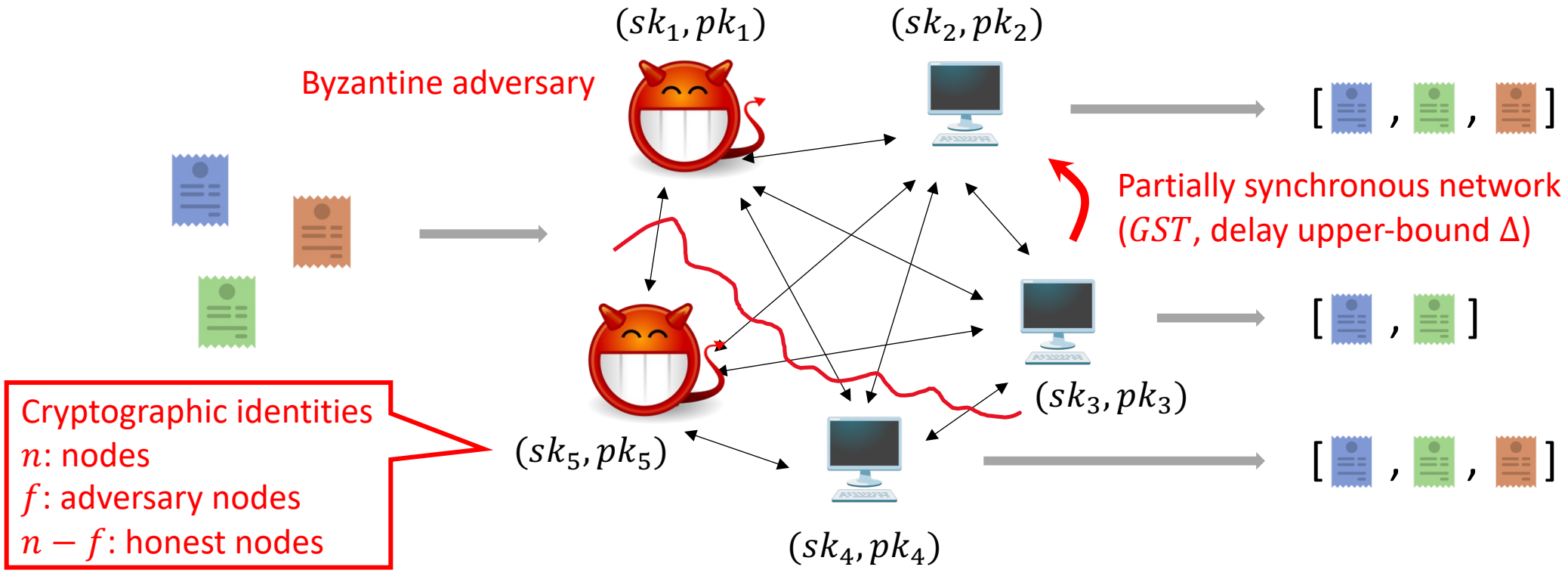
- Consider v with $3\Delta v \geq GST + \Delta$ and honest leader.
- Prop… …ll hone…
- All h… → A…
- → Honest proposal b gets confirmed.

For pedagogy, only considered conflicting blocks from same view. Extension to conflicting blocks from other views requires reasoning with vote-1, vote-2, and lock.

Safety:

- Only one vote-1 per honest node. Assume $n > 3f$.
- For contradiction, suppose both A and B confirmed …
- → At least $n - 2f > f$ nodes 1-voted both A and B.
- → An honest node 1-voted both A and B.
- Contradiction! → No conflicting blocks confirmed!

SMR Example 2: Permissioned PBFT-Style Consensus



Theorem

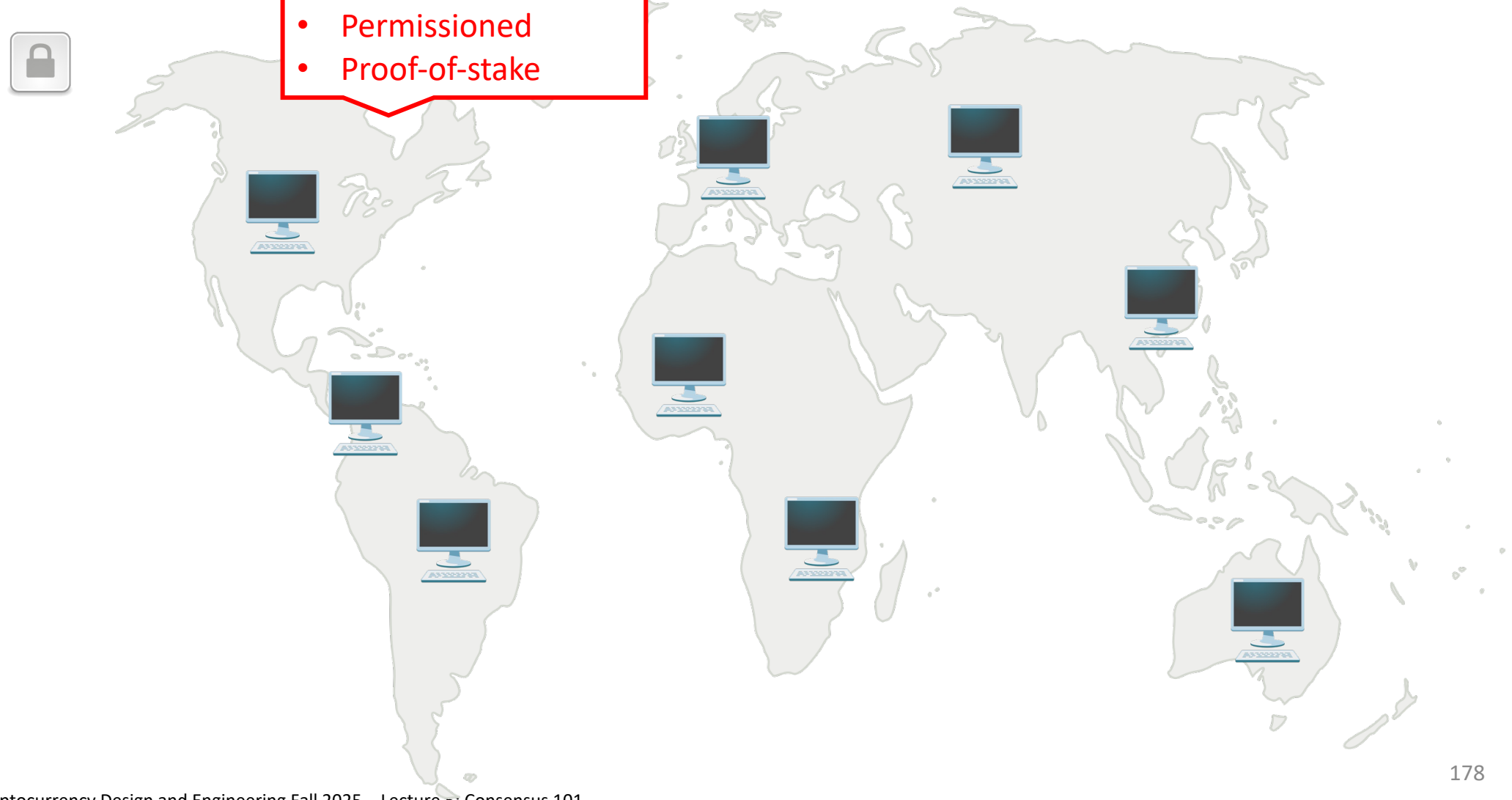
PBFT-style consensus is safe and live iff $f < n/3$, under partial synchrony.

SMR Example 3: Flexible Consensus

Combination of **Example 1: Nakamoto consensus** and **Example 2: PBFT-style consensus**

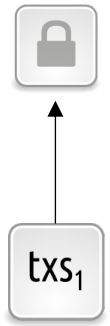
Nakamoto Consensus: Dynamically Available

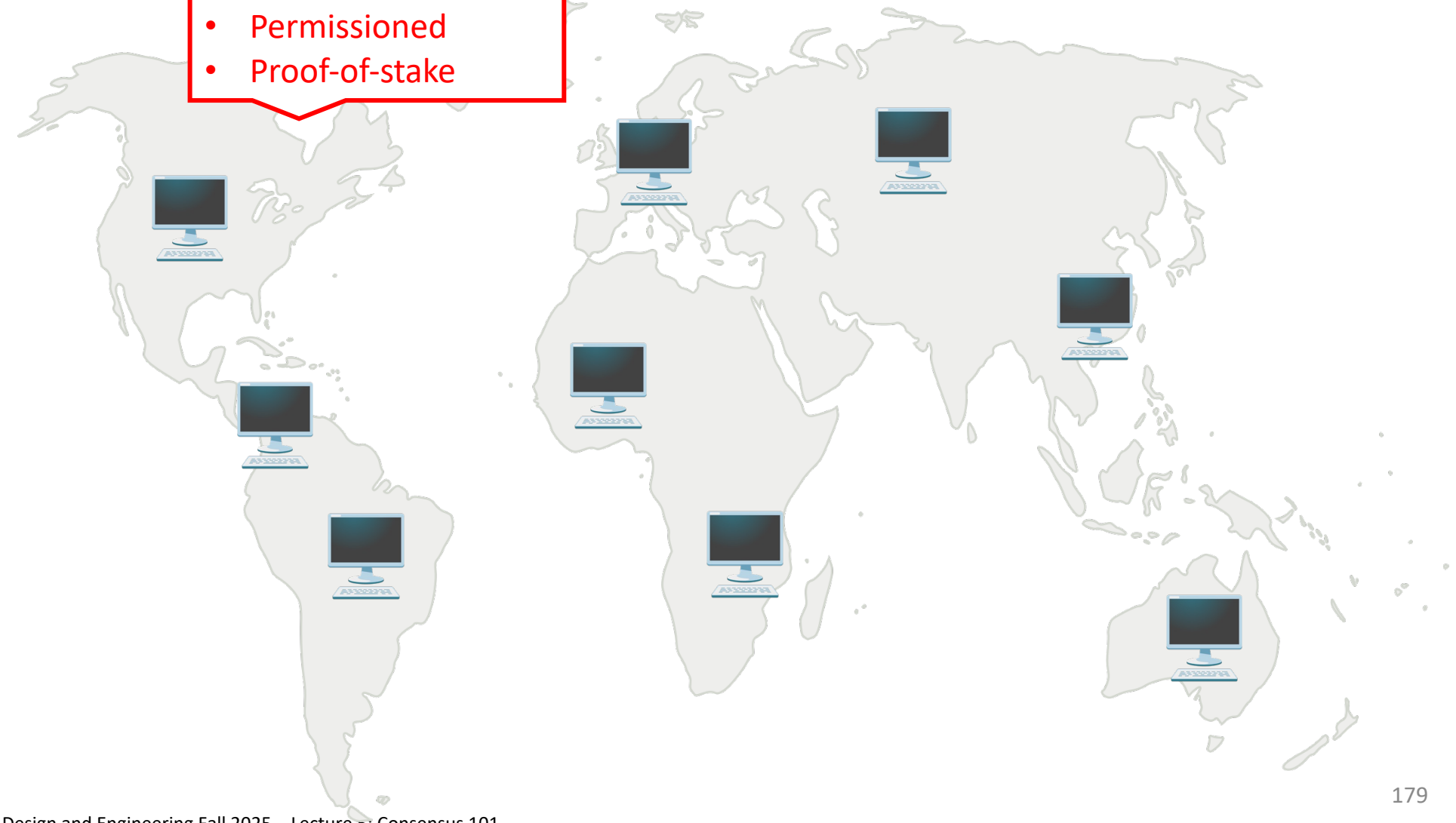
- Proof-of-work
- Permissioned
- Proof-of-stake



Nakamoto Consensus: Dynamically Available

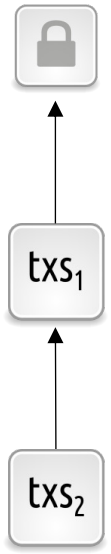
- Proof-of-work
- Permissioned
- Proof-of-stake

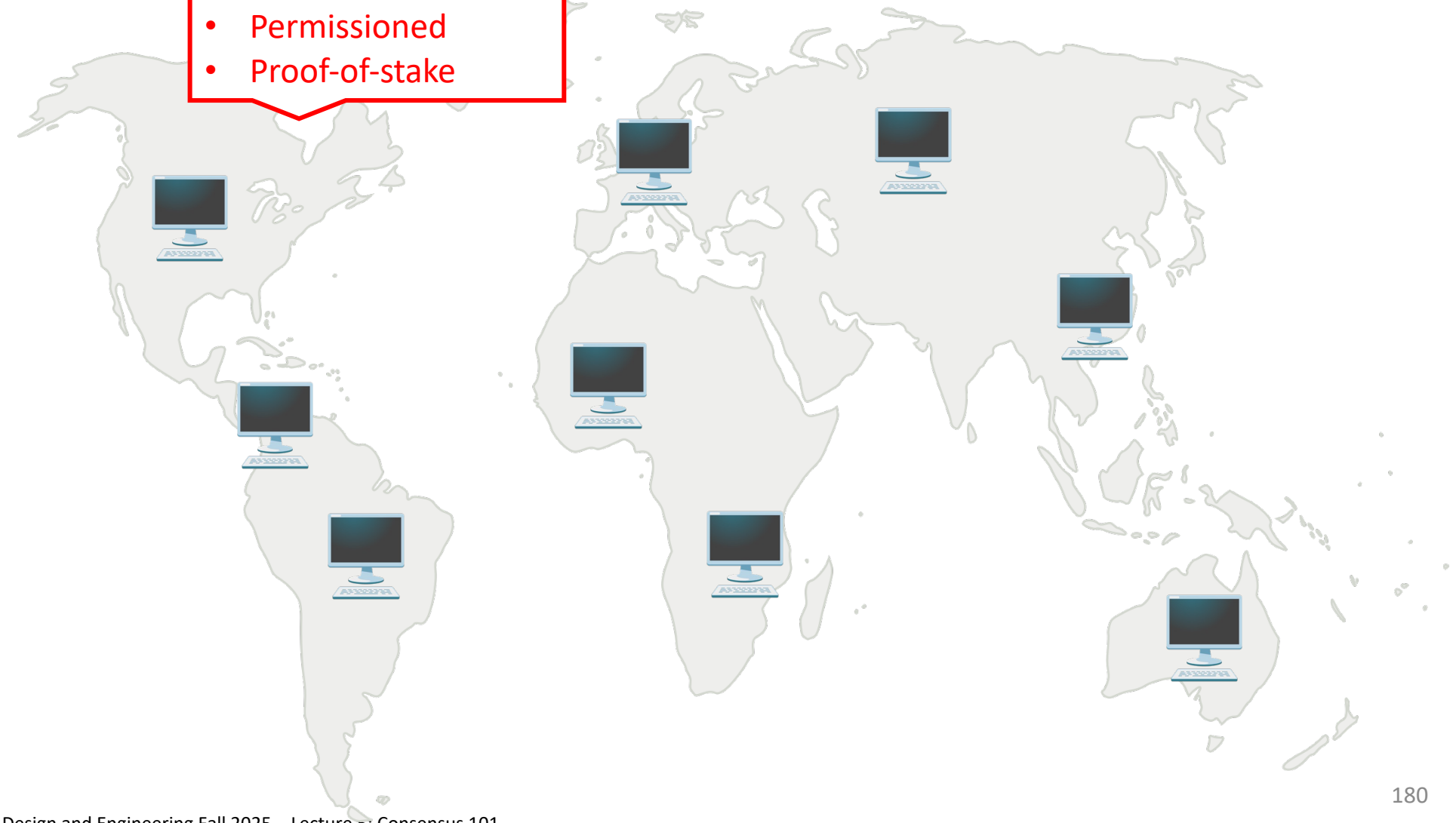




Nakamoto Consensus: Dynamically Available

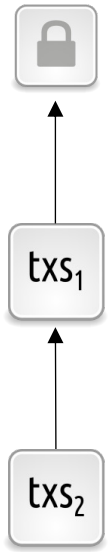
- Proof-of-work
- Permissioned
- Proof-of-stake





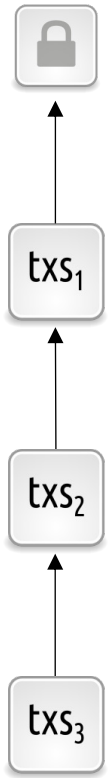
Nakamoto Consensus: Dynamically Available

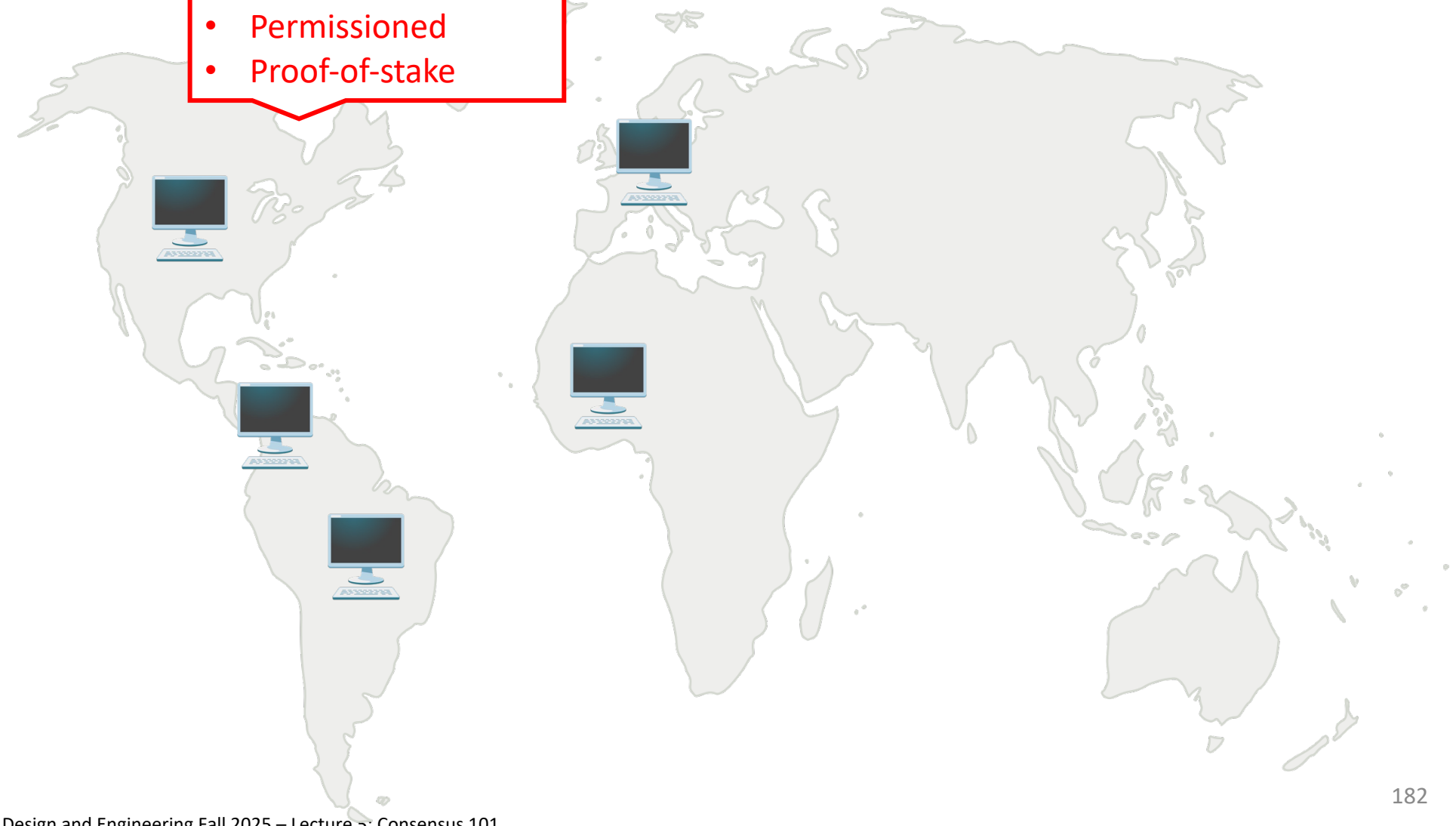
- Proof-of-work
- Permissioned
- Proof-of-stake



Nakamoto Consensus: Dynamically Available

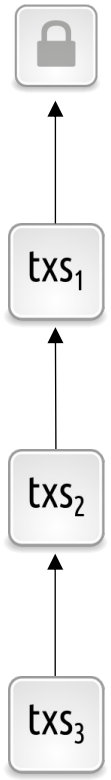
- Proof-of-work
- Permissioned
- Proof-of-stake

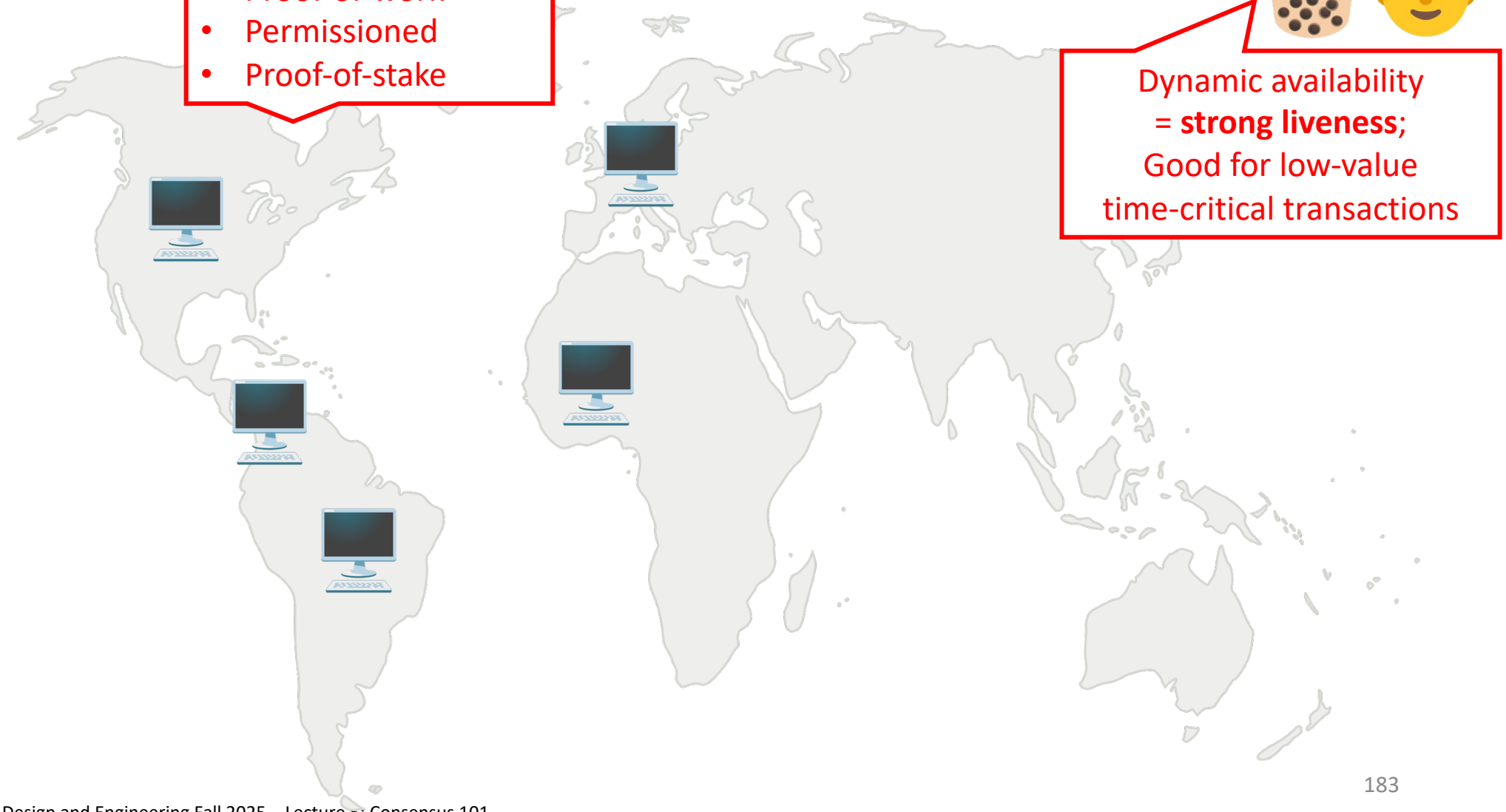




Nakamoto Consensus: Dynamically Available

- Proof-of-work
- Permissioned
- Proof-of-stake

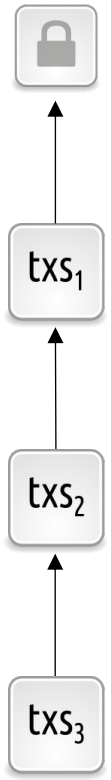


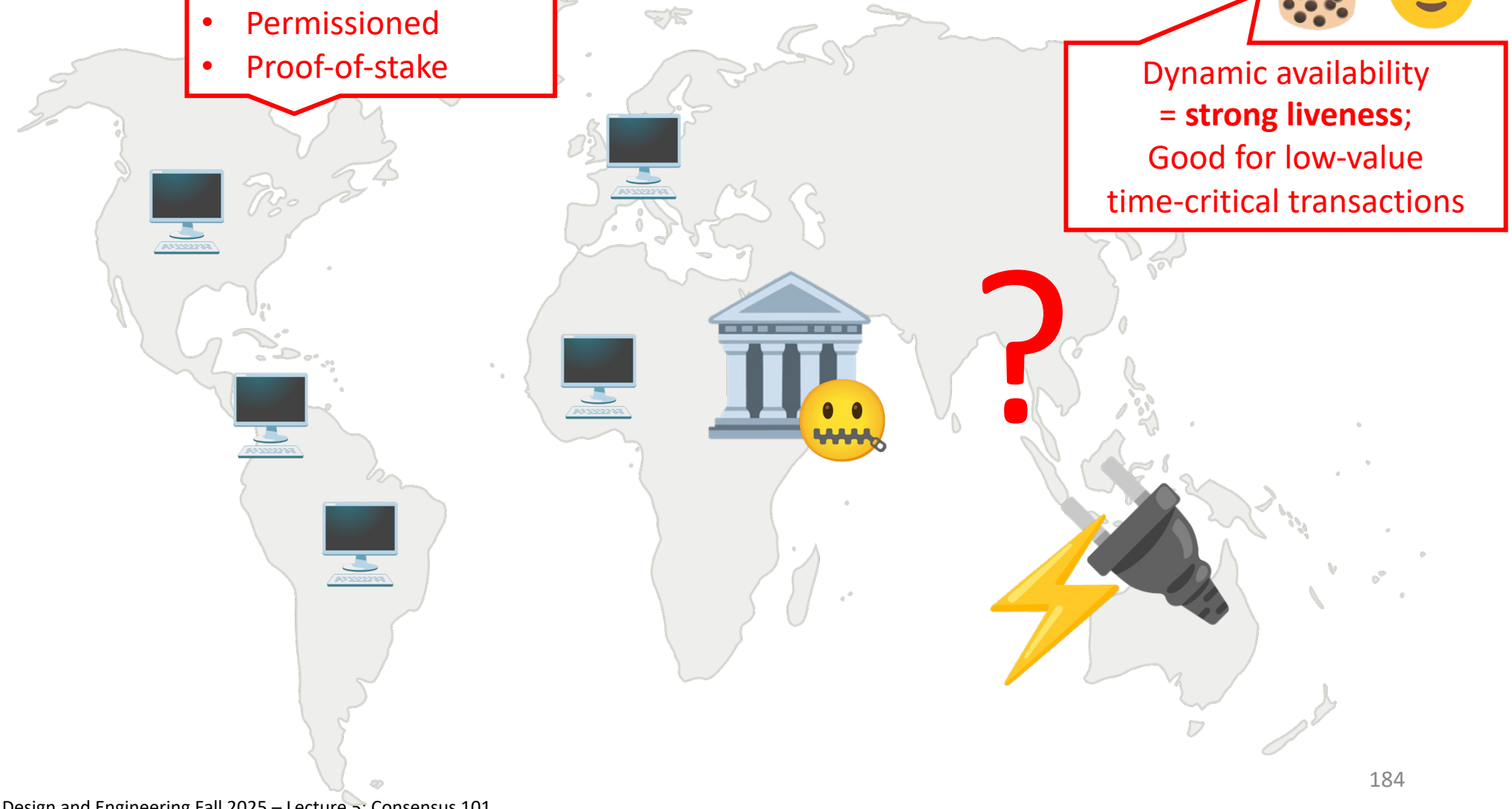


Dynamic availability
= **strong liveness**;
Good for low-value
time-critical transactions

Nakamoto Consensus: Dynamically Available

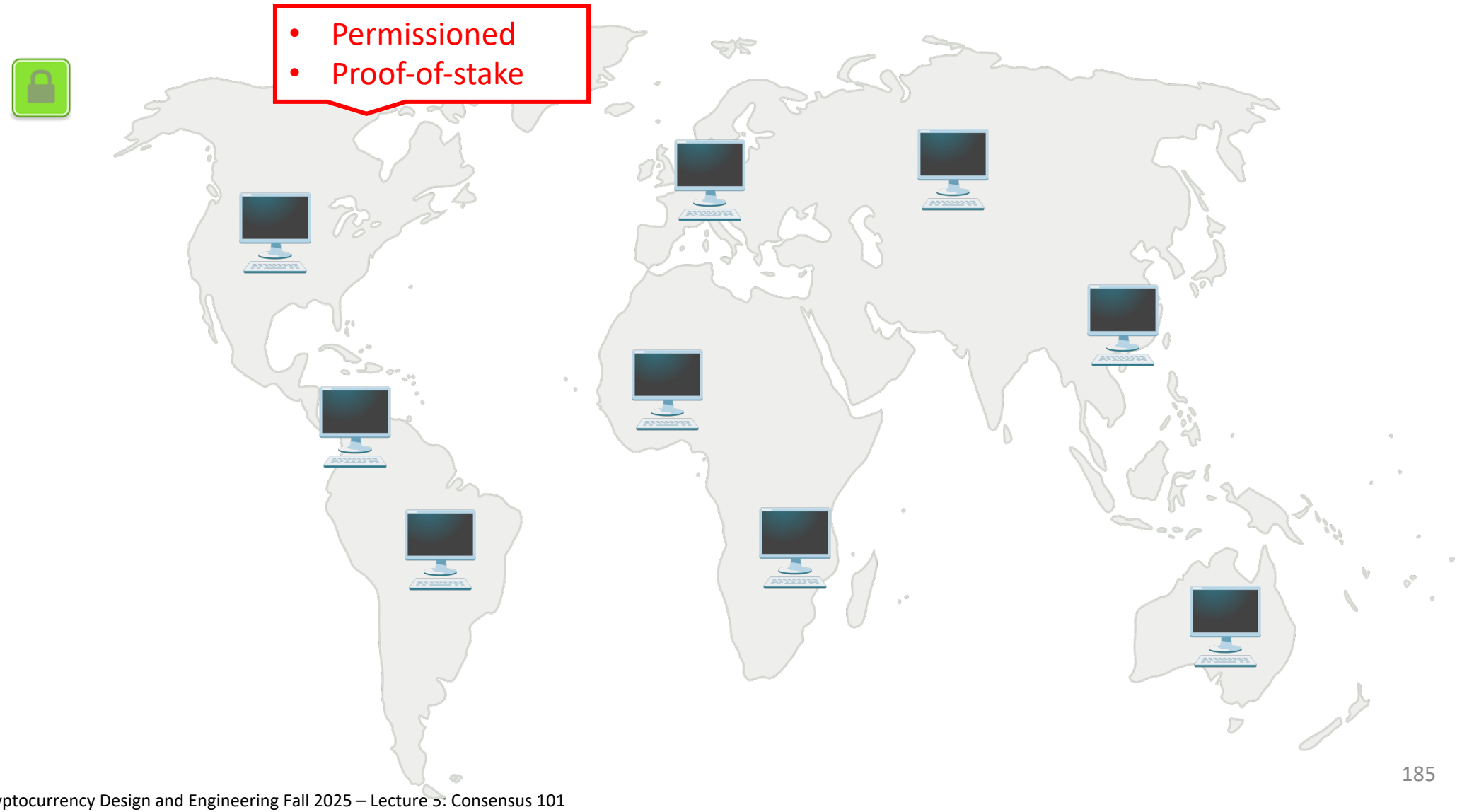
- Proof-of-work
- Permissioned
- Proof-of-stake



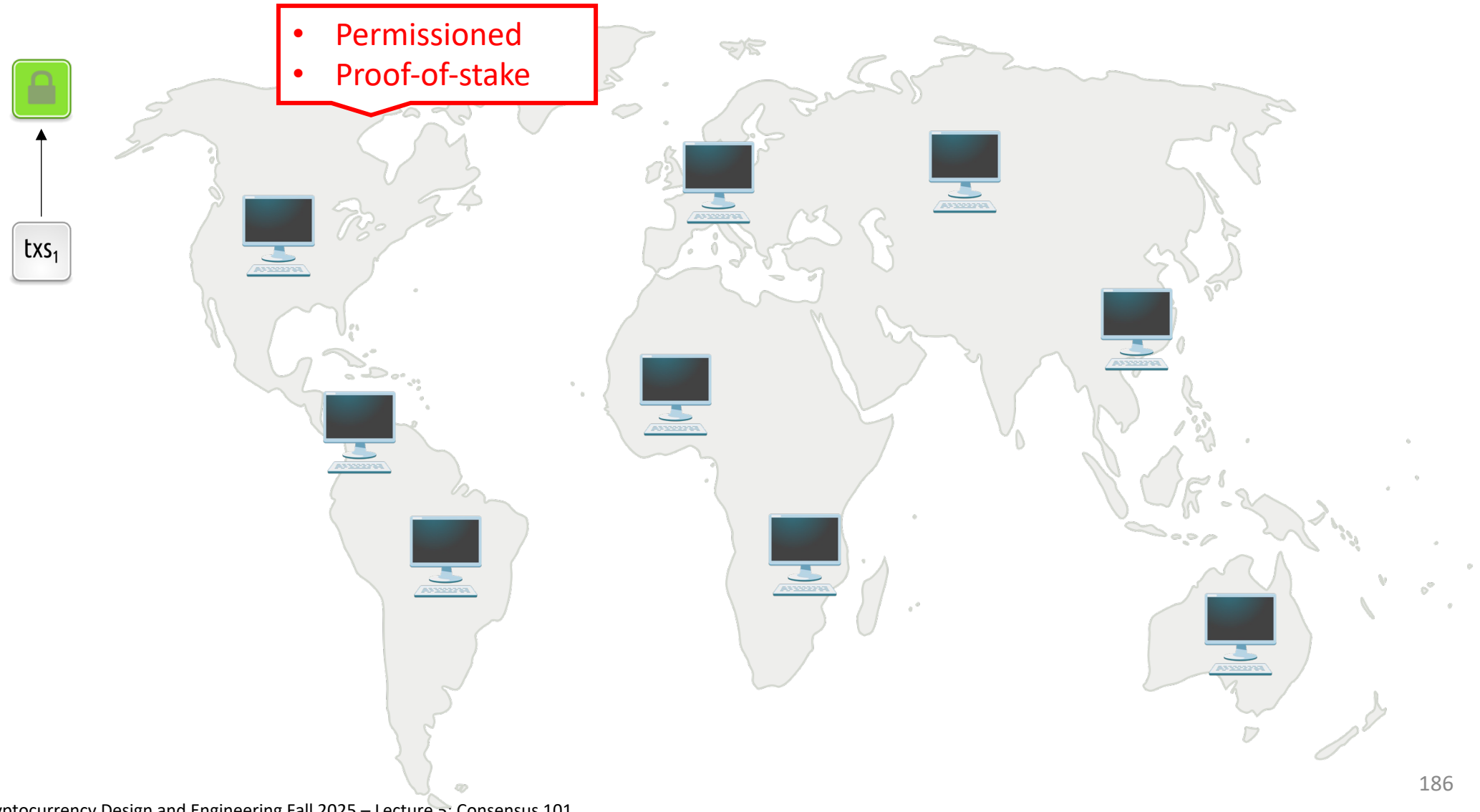


Dynamic availability
= **strong liveness**;
Good for low-value
time-critical transactions

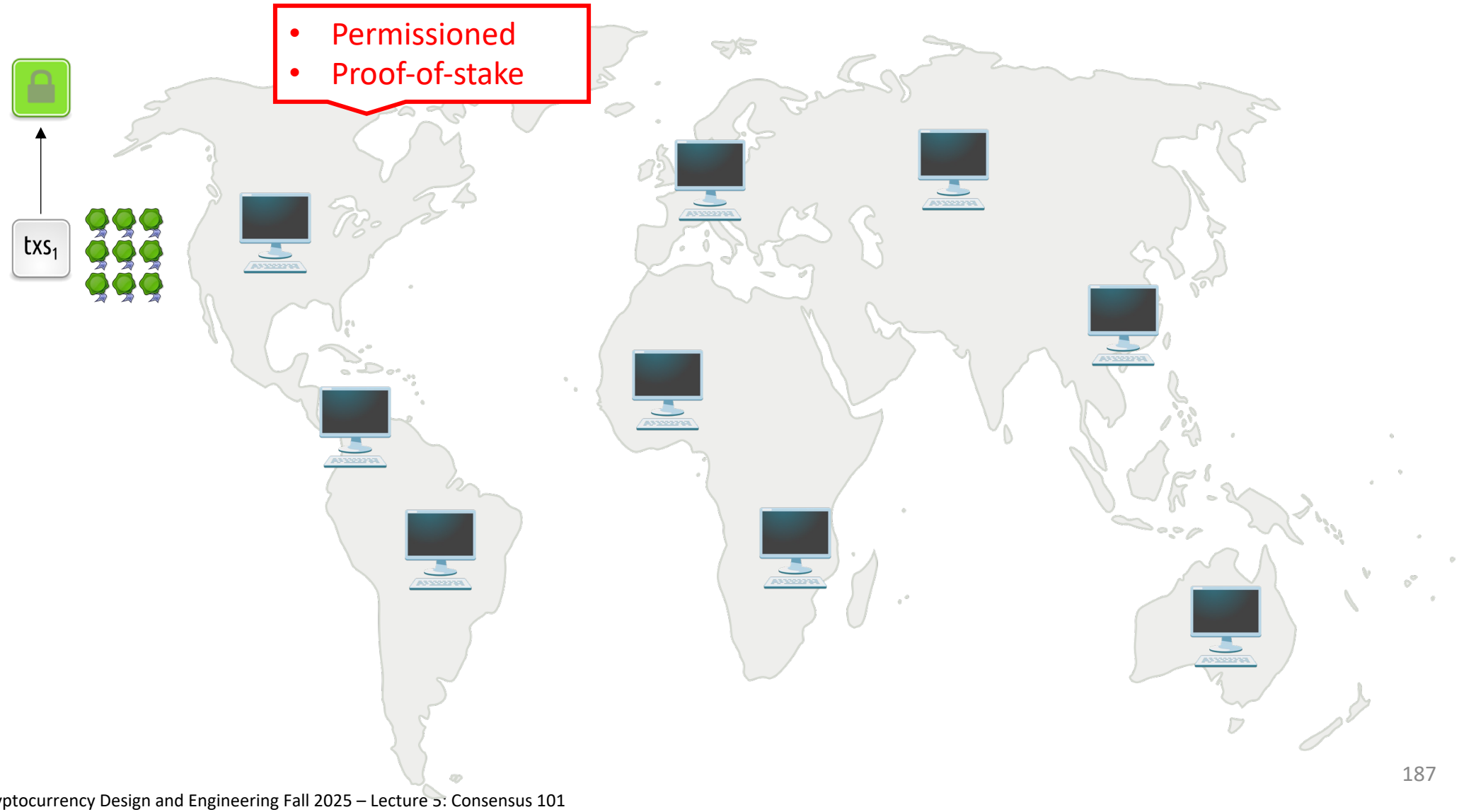
PBFT-Style Consensus: Accountably Safe



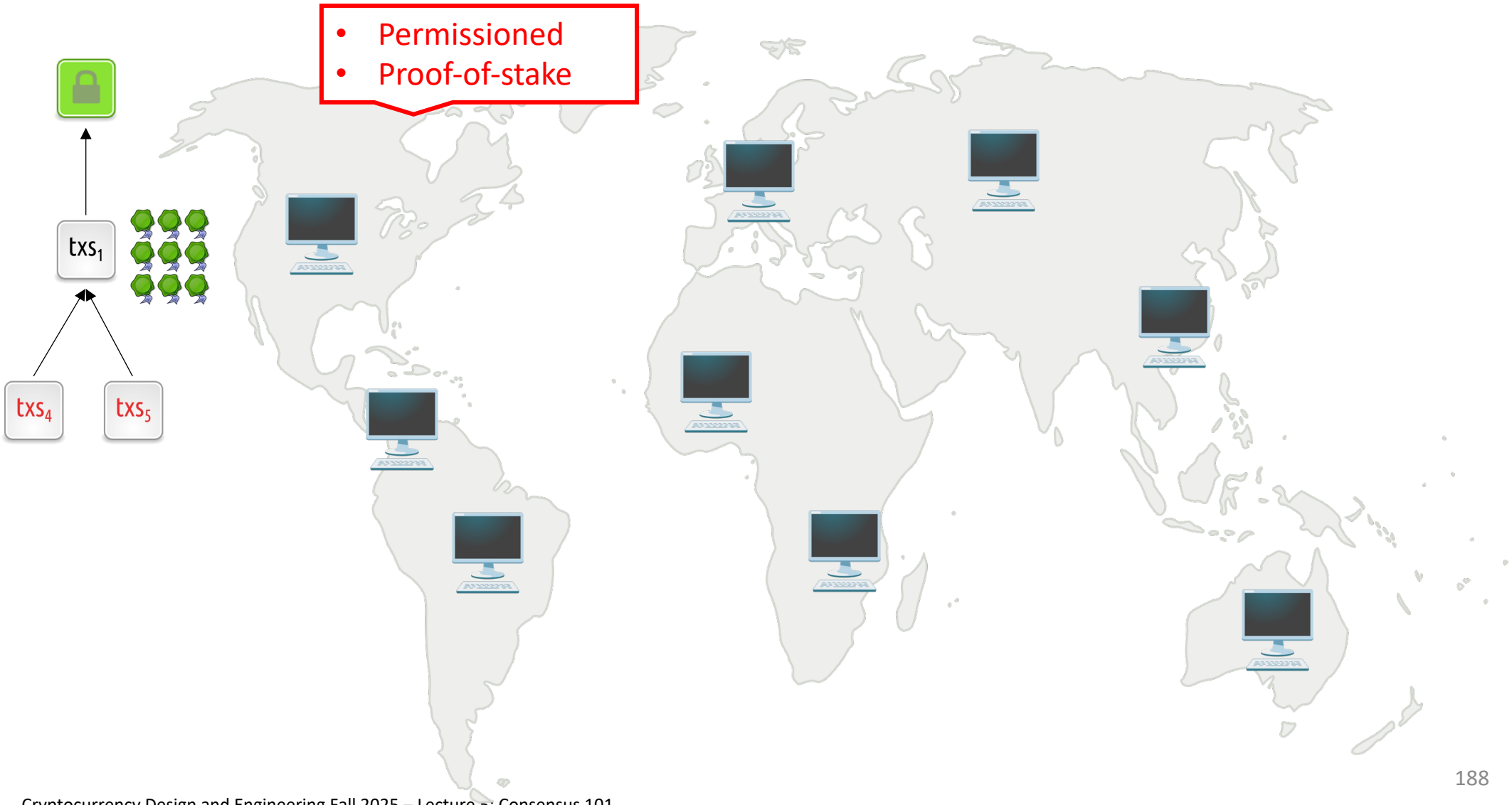
PBFT-Style Consensus: Accountably Safe



PBFT-Style Consensus: Accountably Safe

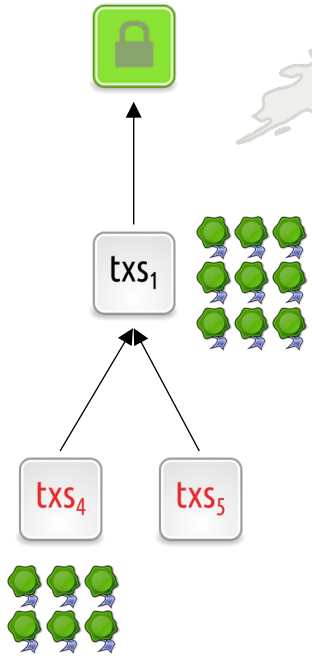


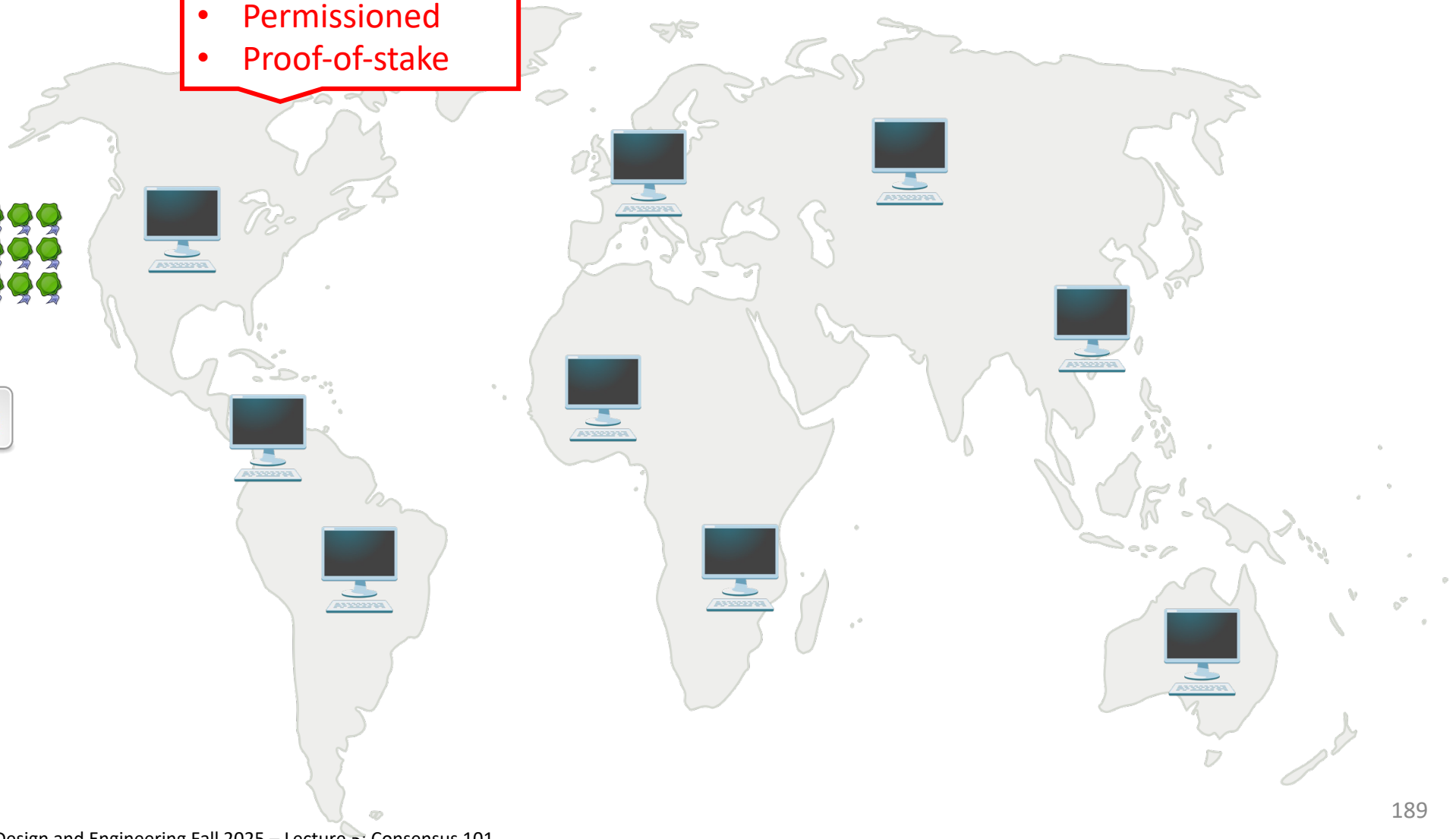
PBFT-Style Consensus: Accountably Safe



PBFT-Style Consensus: Accountably Safe

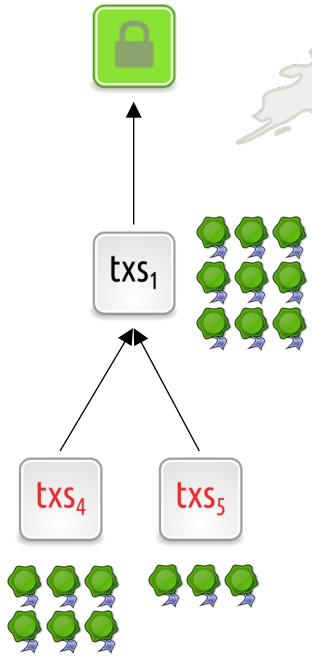
- Permissioned
- Proof-of-stake

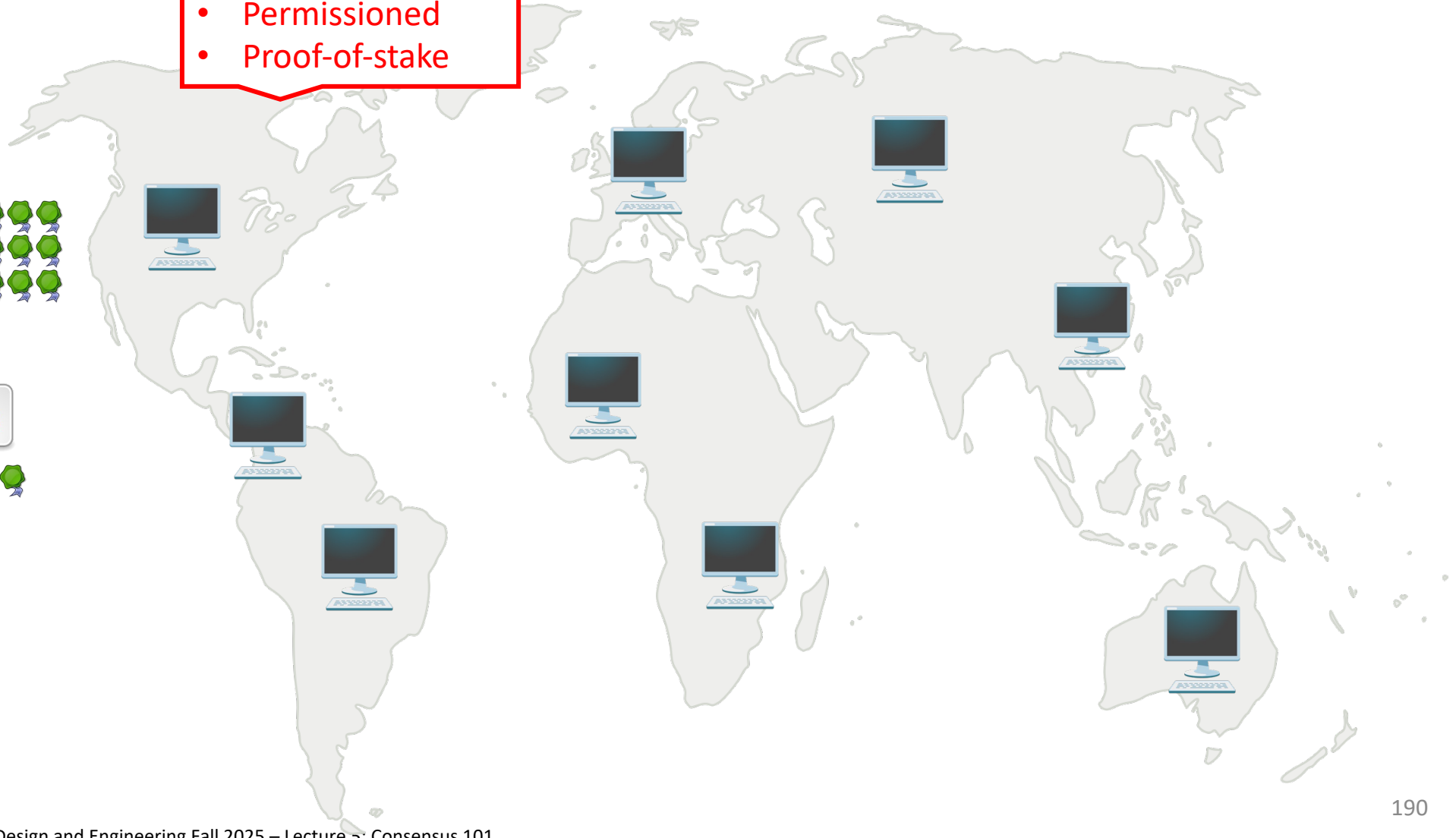




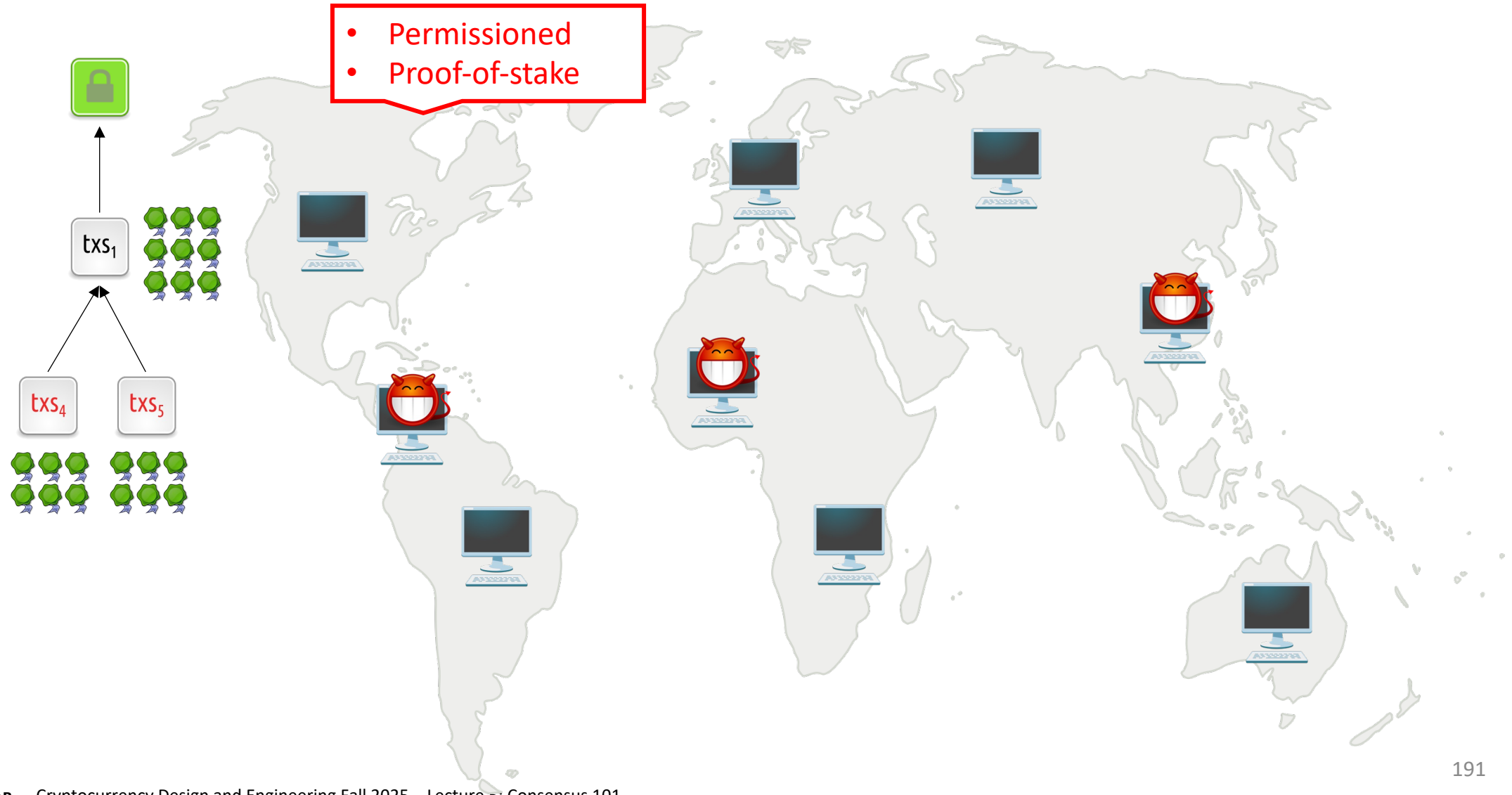
PBFT-Style Consensus: Accountably Safe

- Permissioned
- Proof-of-stake



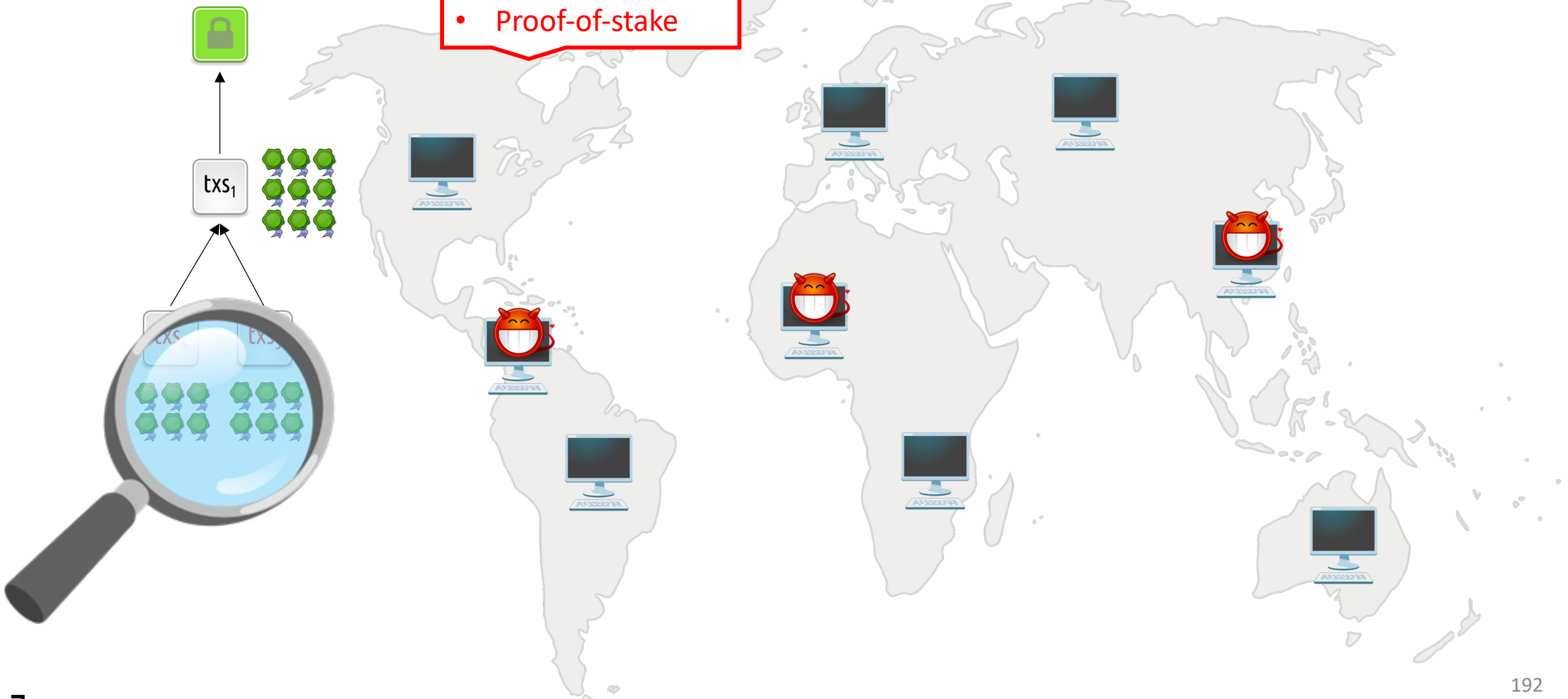


PBFT-Style Consensus: Accountably Safe

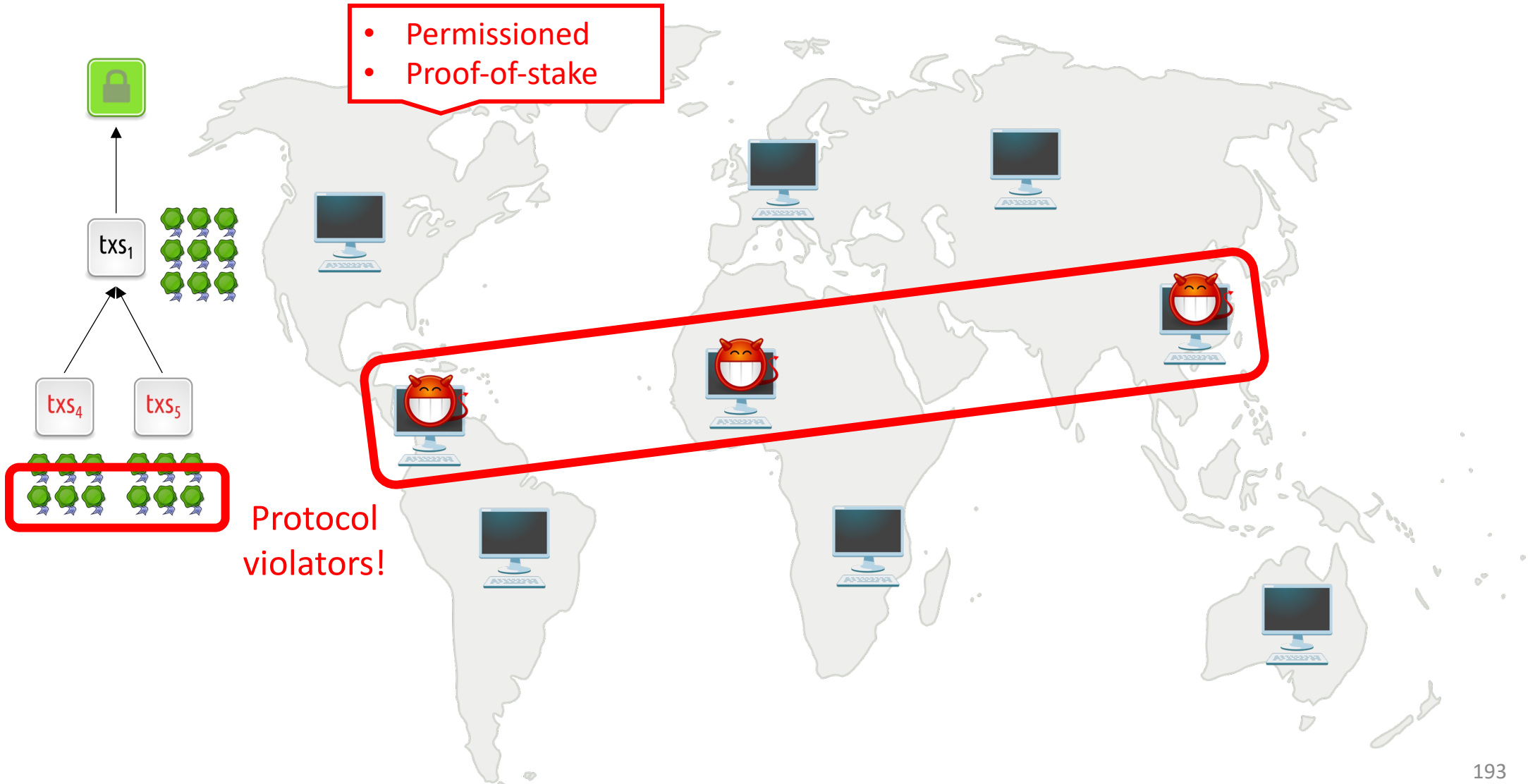


PBFT-Style Consensus: Accountably Safe

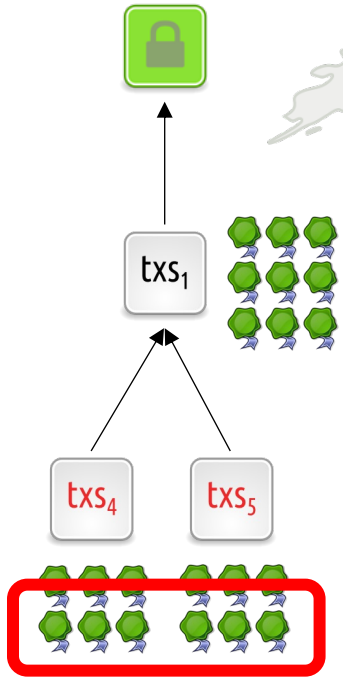
- Permissioned
- Proof-of-stake



PBFT-Style Consensus: Accountably Safe



PBFT-Style Consensus: Accountably Safe

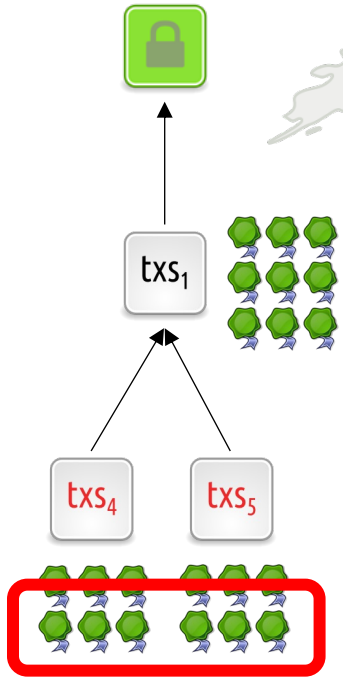


- Permissioned
- Proof-of-stake

Protocol violators!

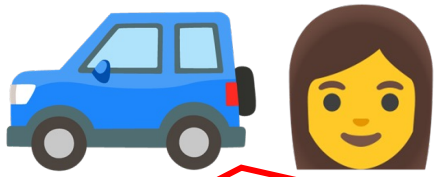
Accountable safety
= **strong safety**;
Good for high-value
less-urgent transactions

PBFT-Style Consensus: Accountably Safe



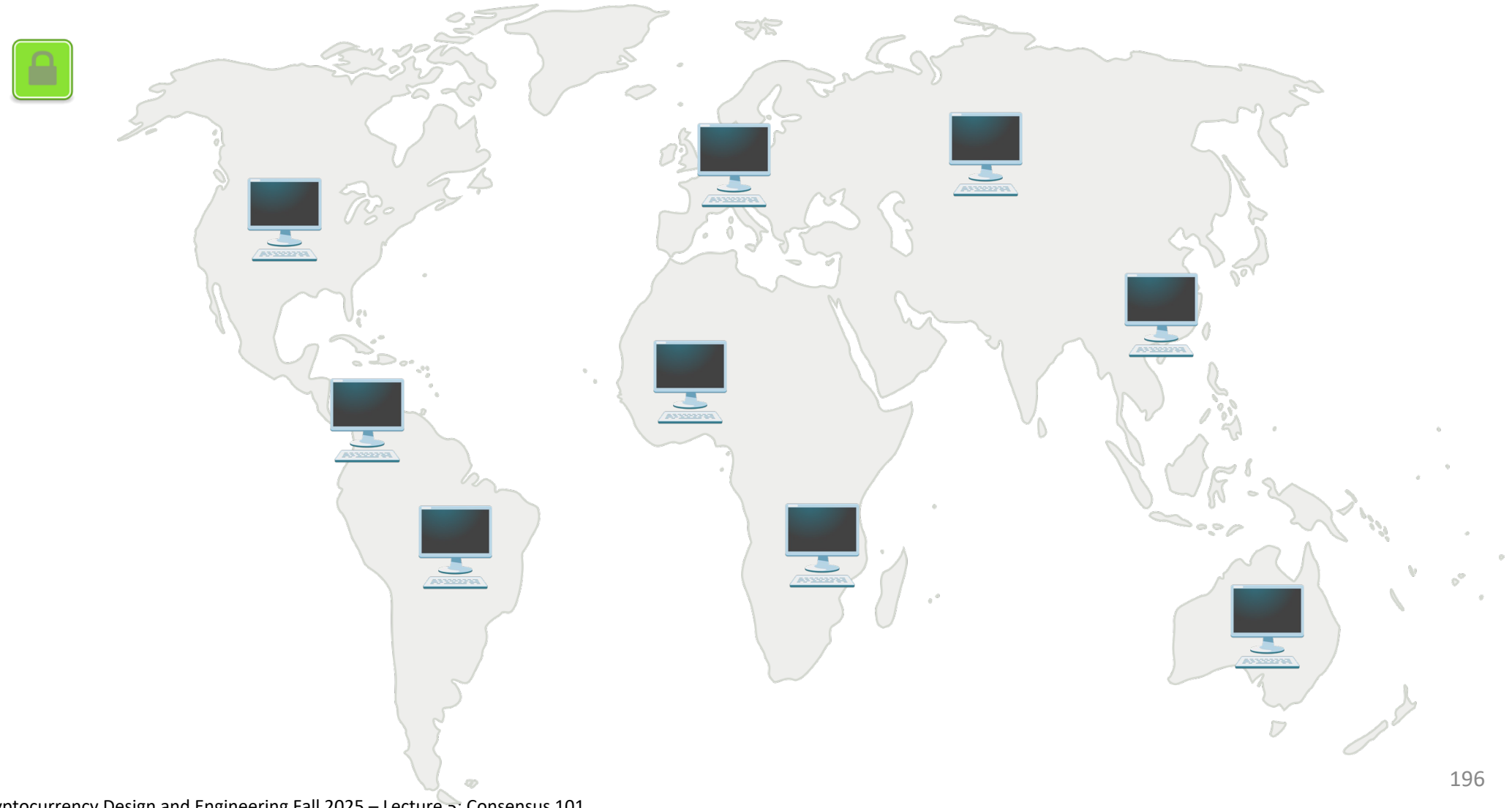
- Permissioned
- Proof-of-stake

Protocol violators!

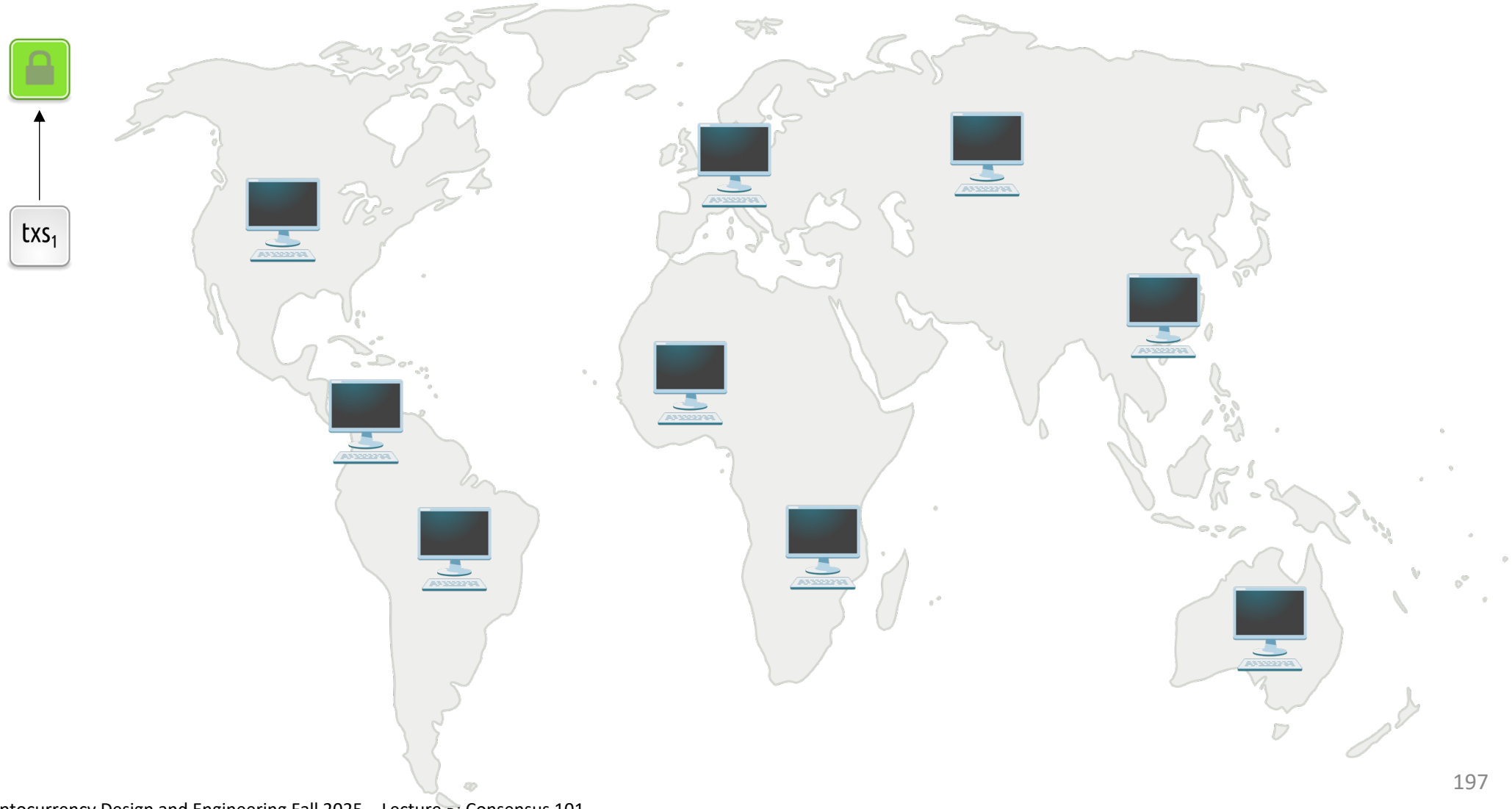


Accountable safety
= **strong safety**;
Good for high-value
less-urgent transactions

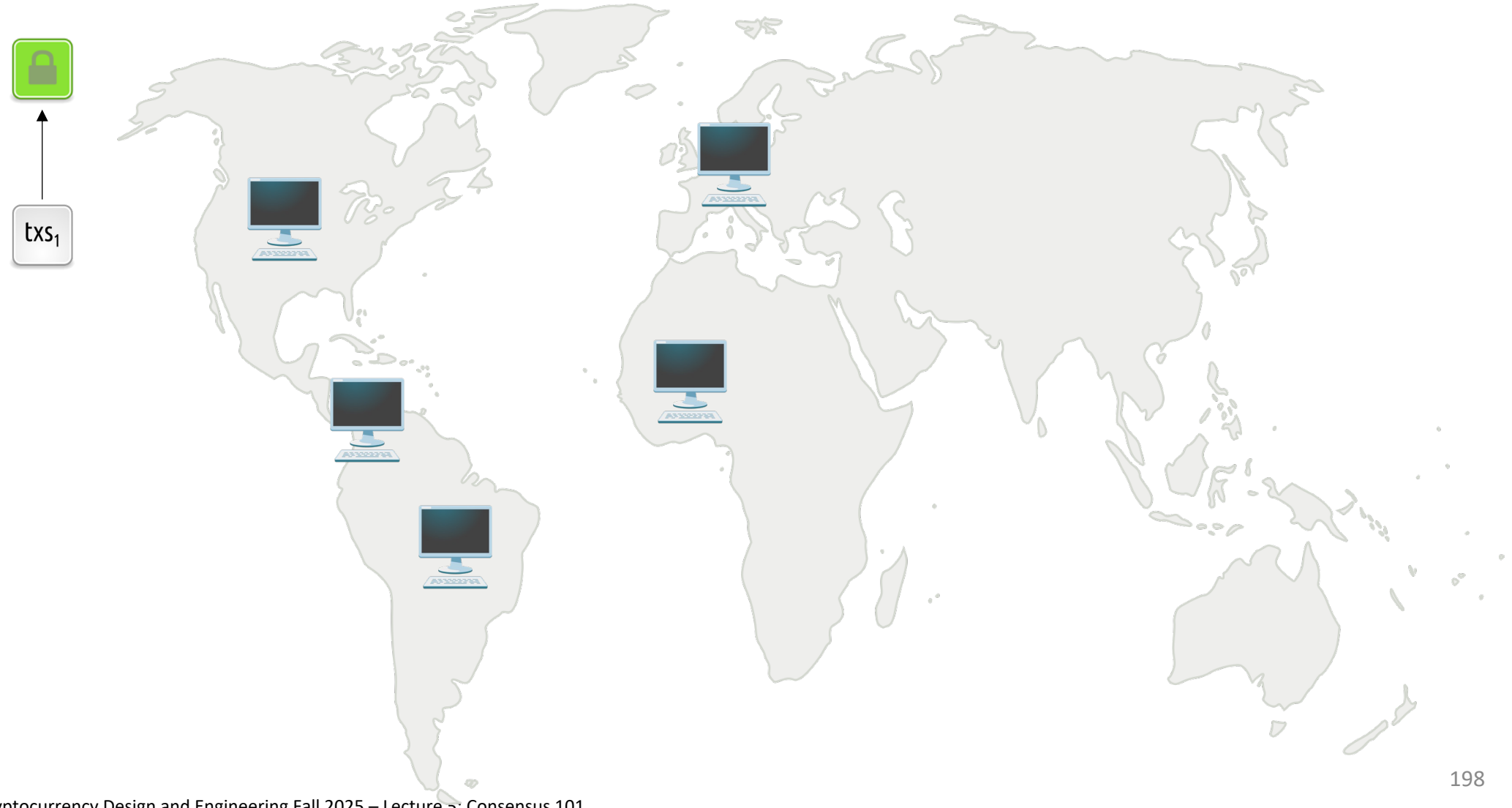
PBFT-Style Consensus: Not Dynamically Available



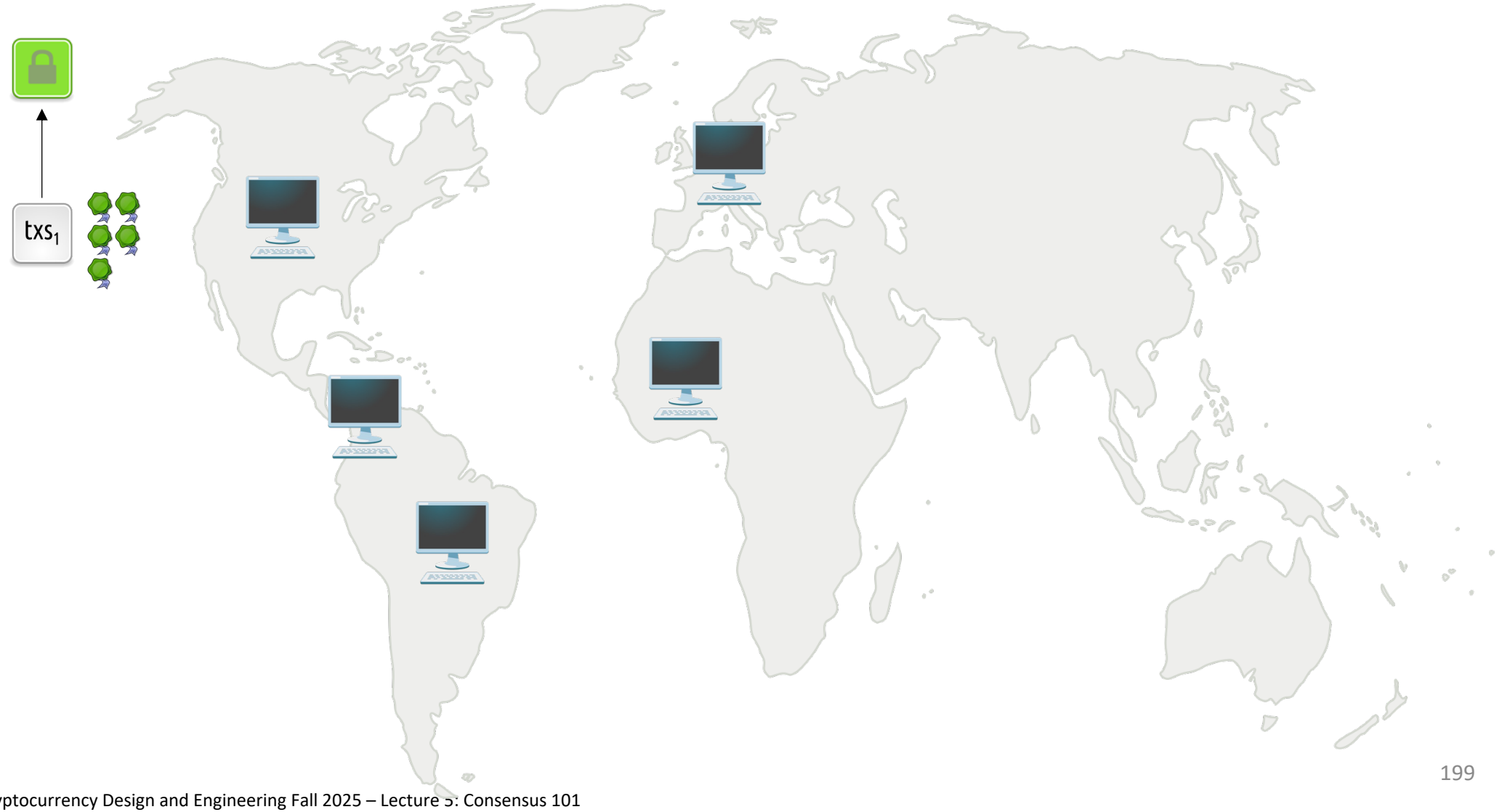
PBFT-Style Consensus: Not Dynamically Available



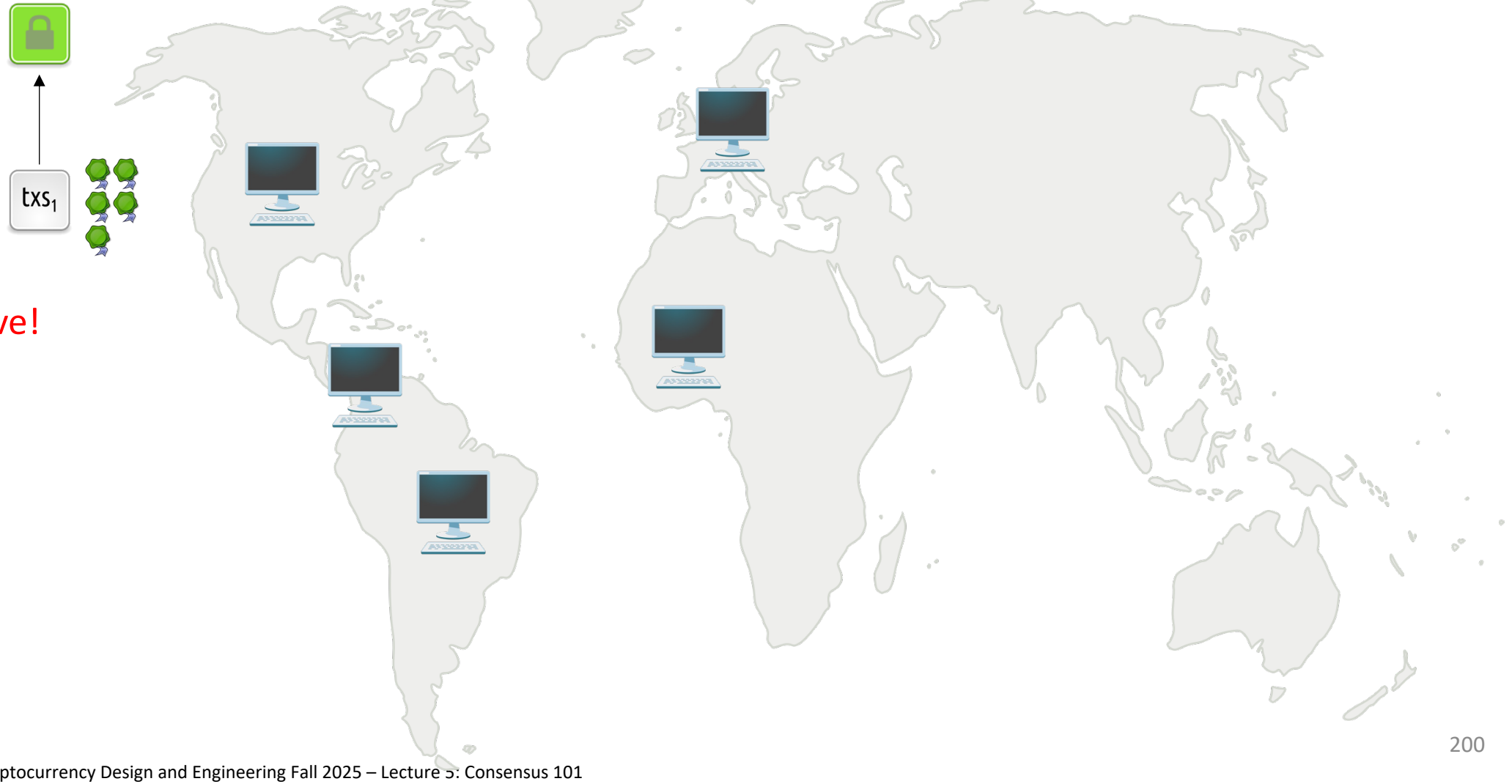
PBFT-Style Consensus: Not Dynamically Available



PBFT-Style Consensus: Not Dynamically Available



PBFT-Style Consensus: Not Dynamically Available



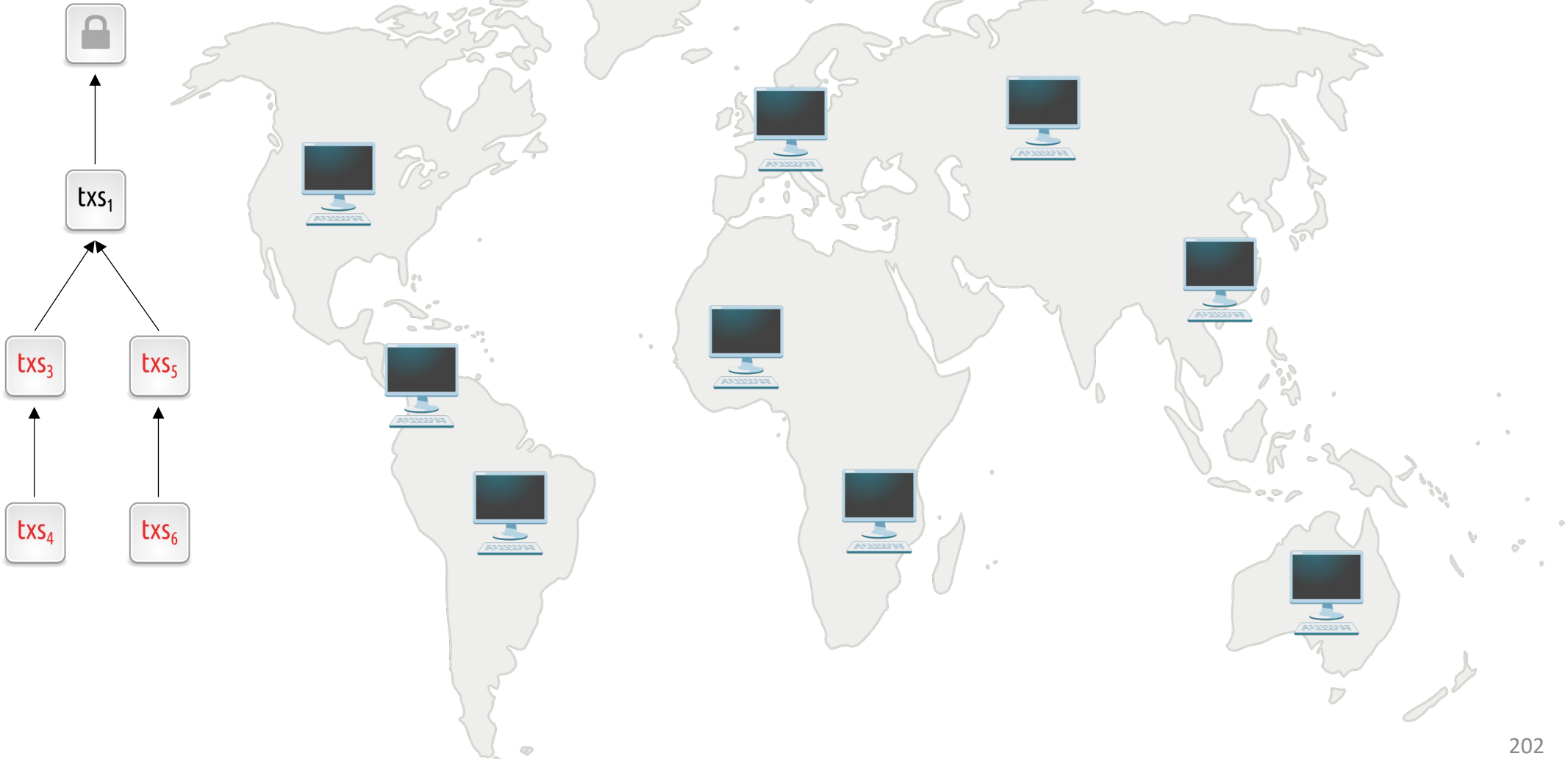
Not live!

Gilbert, Lynch 2002; Lewis-Pye, Roughgarden 2020; Neu et al. 2024

Nakamoto Consensus: Not Accountably Safe

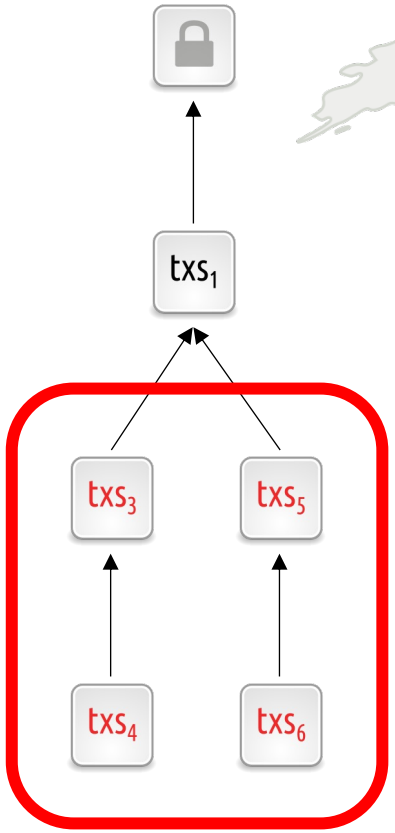
Gilbert, Lynch 2002; Lewis-Pye, Roughgarden 2020; Neu et al. 2024

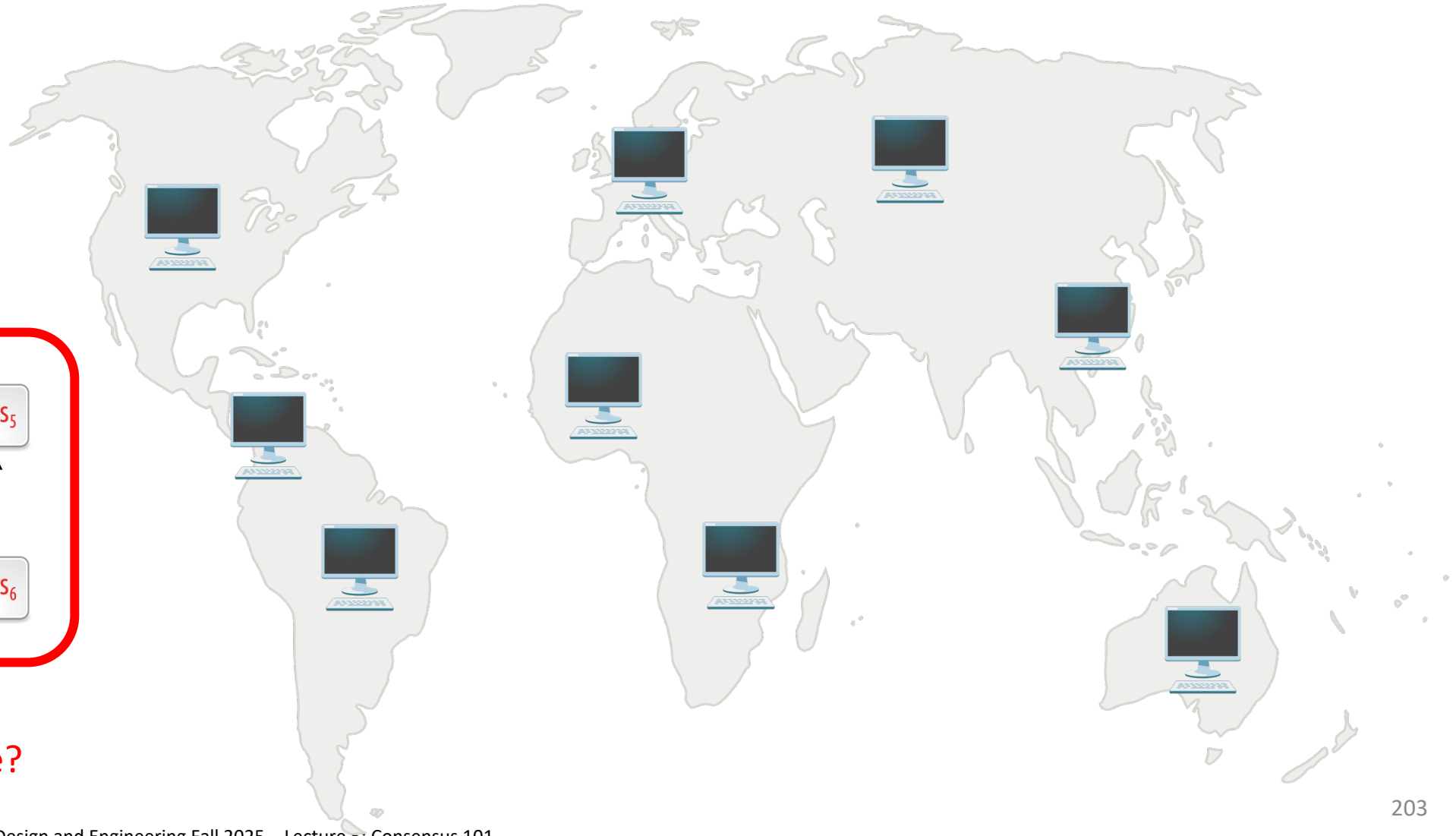
Nakamoto Consensus: Not Accountably Safe



Gilbert, Lynch 2002; Lewis-Pye, Roughgarden 2020; Neu et al. 2024

Nakamoto Consensus: Not Accountably Safe

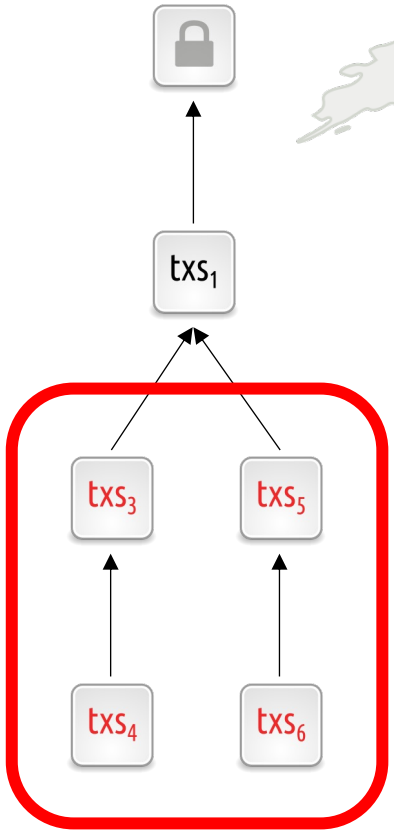




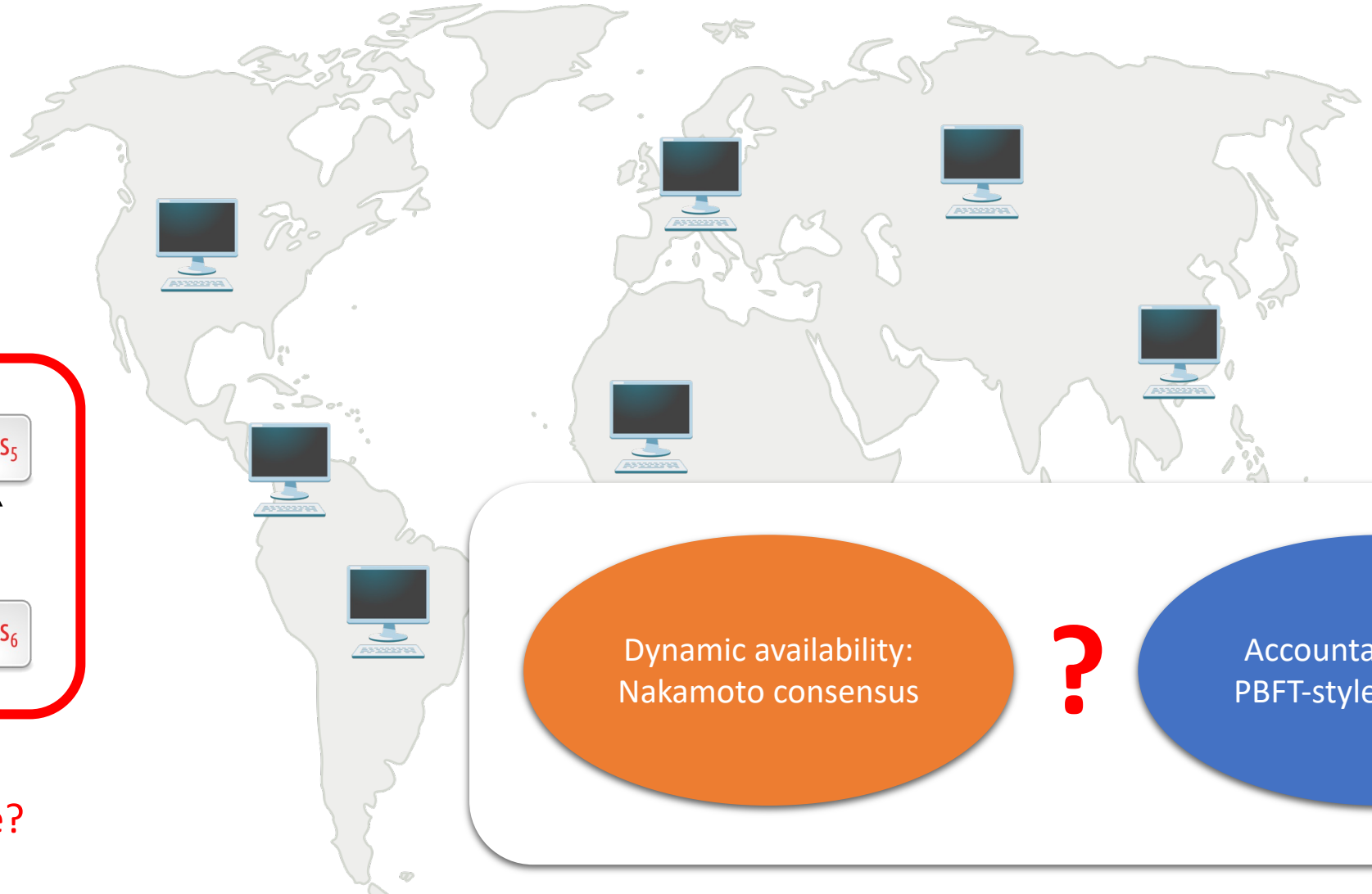
Safety violation.
Who is to blame?

Gilbert, Lynch 2002; Lewis-Pye, Roughgarden 2020; Neu et al. 2024

Nakamoto Consensus: Not Accountably Safe



Safety violation.
Who is to blame?



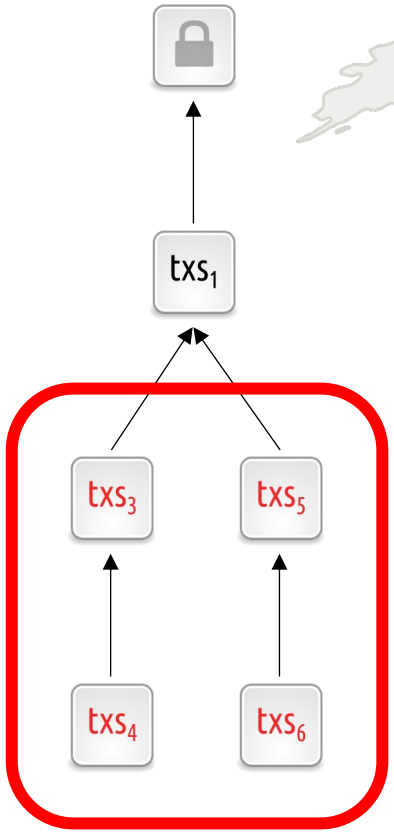
Dynamic availability:
Nakamoto consensus

?

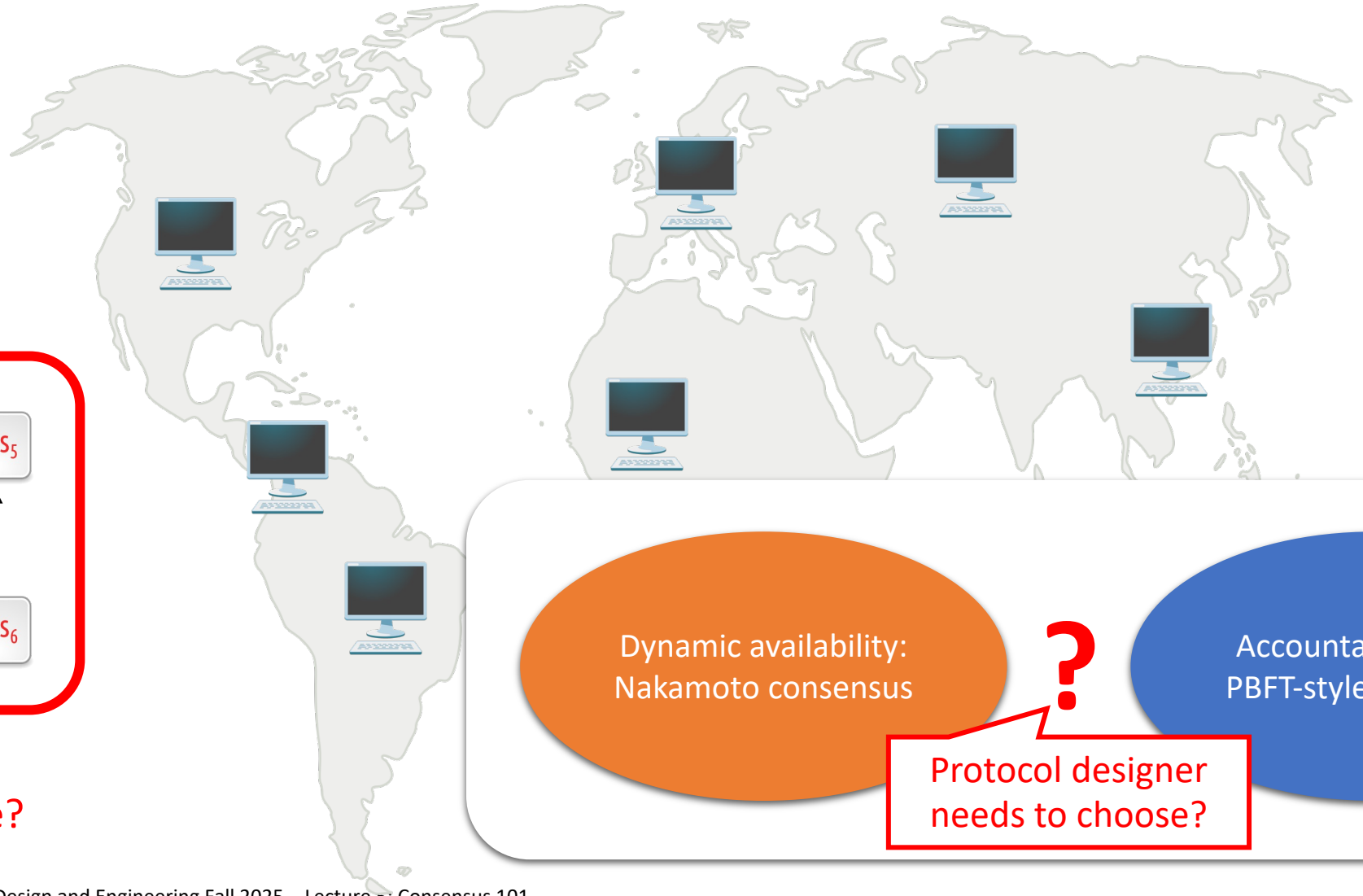
Accountable safety:
PBFT-style consensus

Gilbert, Lynch 2002; Lewis-Pye, Roughgarden 2020; Neu et al. 2024

Nakamoto Consensus: Not Accountably Safe



Safety violation.
Who is to blame?



Dynamic availability:
Nakamoto consensus

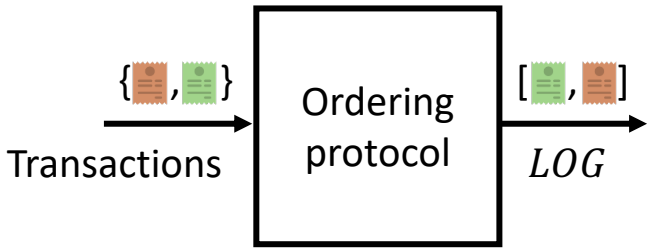
?

Accountable safety:
PBFT-style consensus

Protocol designer
needs to choose?

"Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma", Neu et al. 2021

Dynamic Availability + Accountable Safety



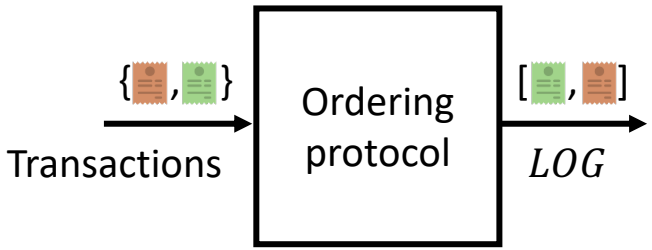
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

Impossible!

"Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma", Neu et al. 2021

Dynamic Availability + Accountable Safety



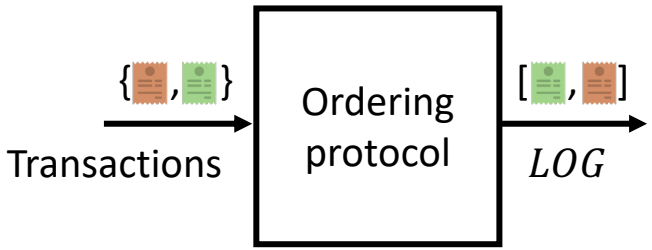
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

Impossible!

"Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma", Neu et al. 2021

Dynamic Availability + Accountable Safety



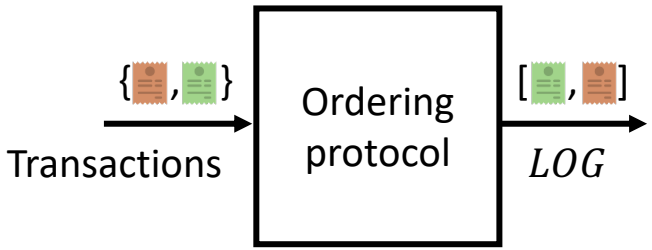
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

Impossible!

"Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma", Neu et al. 2021

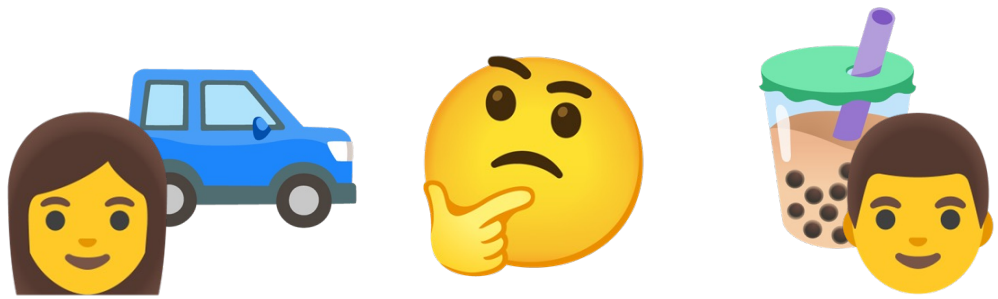
Dynamic Availability + Accountable Safety



Single log: $tx_1, tx_2, tx_3, \ldots$

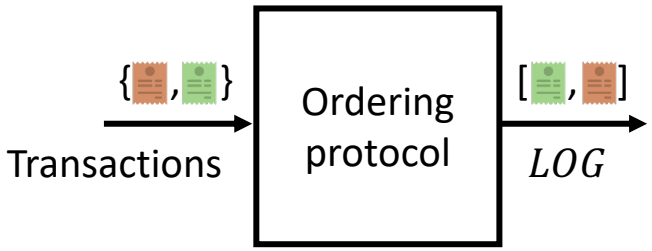
Protocol designer needs to choose?

Impossible!



"Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma", Neu et al. 2021

Dynamic Availability + Accountable Safety

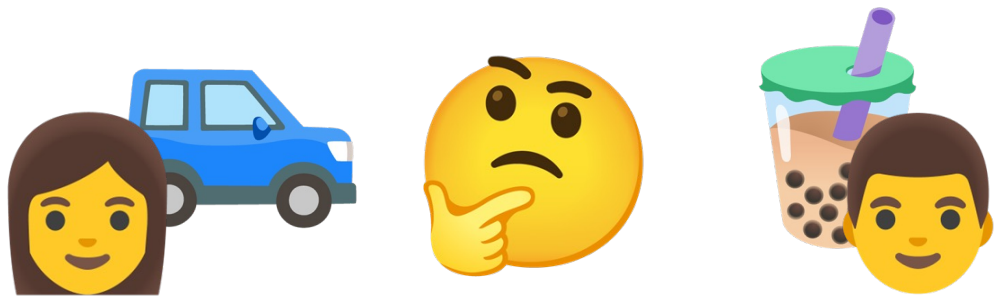


Single log: $tx_1, tx_2, tx_3, \ldots$

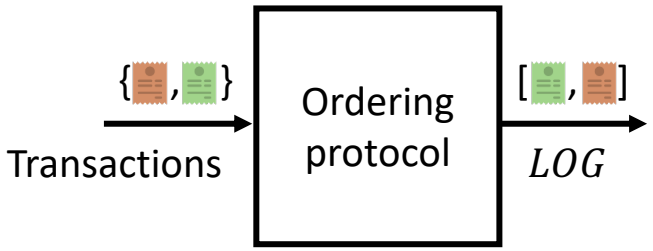
Protocol designer needs to choose?

Impossible!

Available *full* log



Dynamic Availability + Accountable Safety

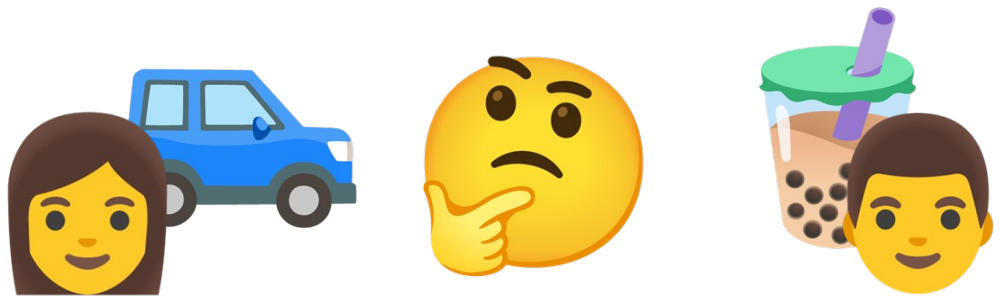


Single log: $tx_1, tx_2, tx_3, \ldots$

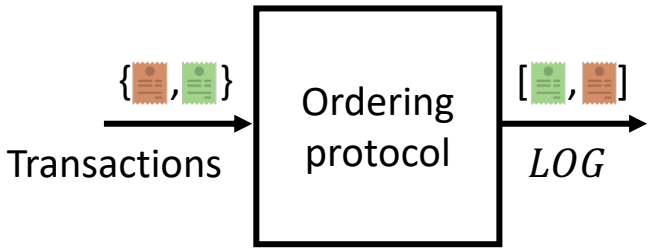
Protocol designer needs to choose?

Impossible!

Accountable *prefix* log | Available *full* log



Dynamic Availability + Accountable Safety

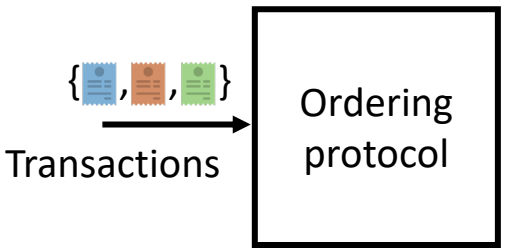


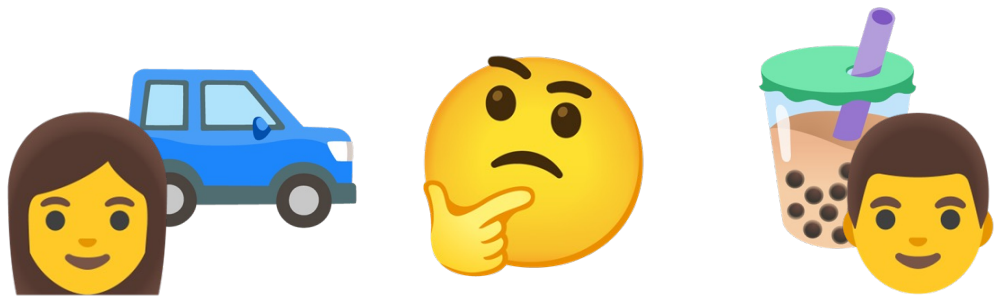
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

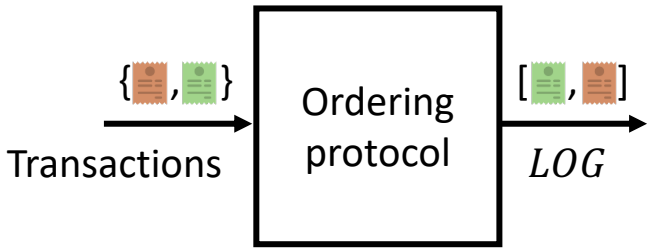
Impossible!

Accountable *prefix* log | Available *full* log





Dynamic Availability + Accountable Safety

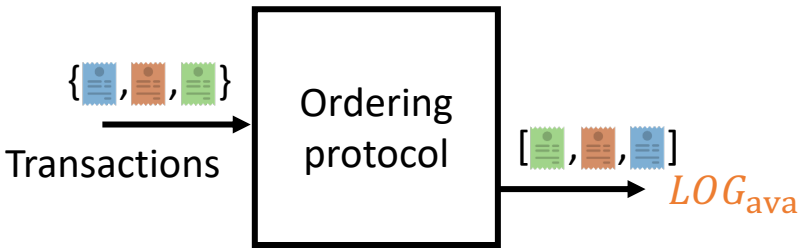


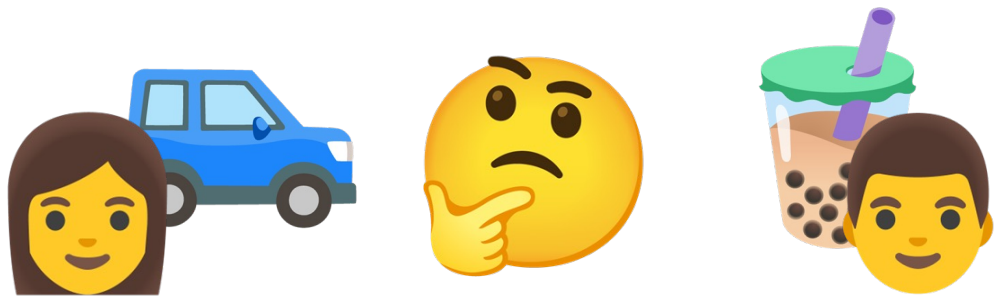
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

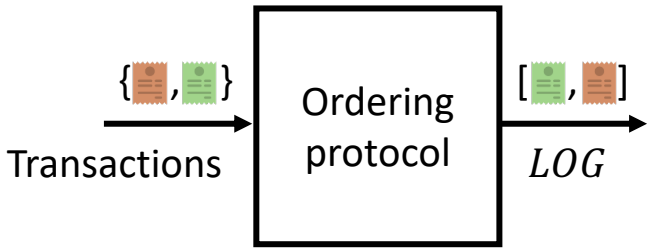
Impossible!

Accountable *prefix* log | Available *full* log





Dynamic Availability + Accountable Safety

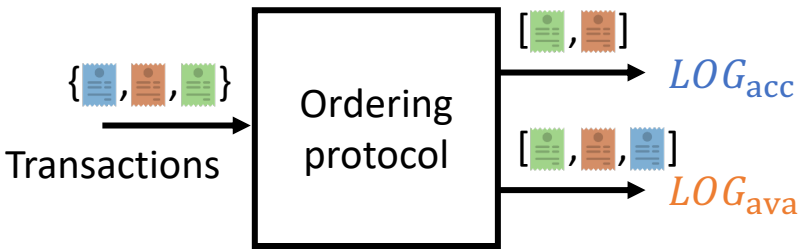


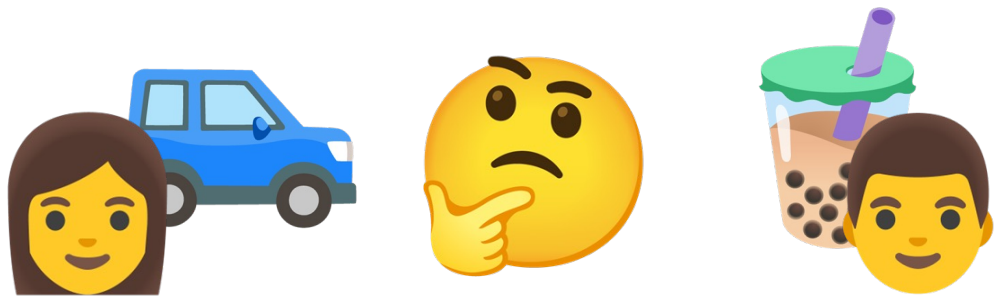
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

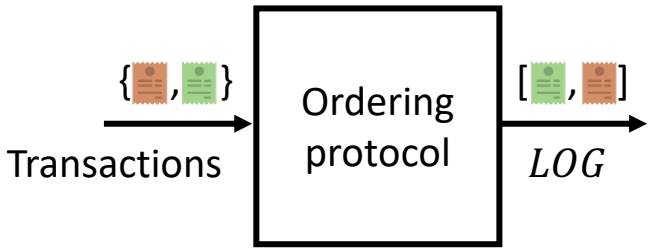
Impossible!

Accountable *prefix* log | Available *full* log





Dynamic Availability + Accountable Safety

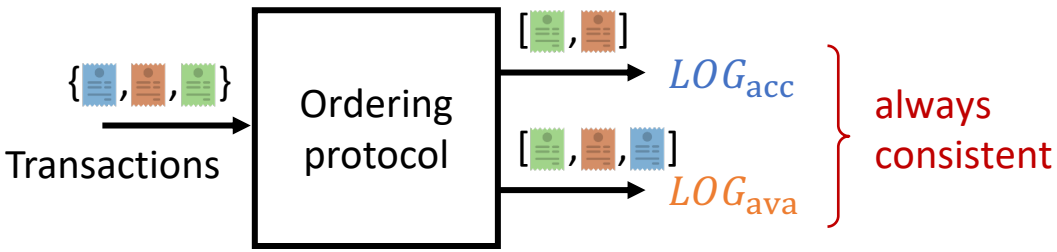


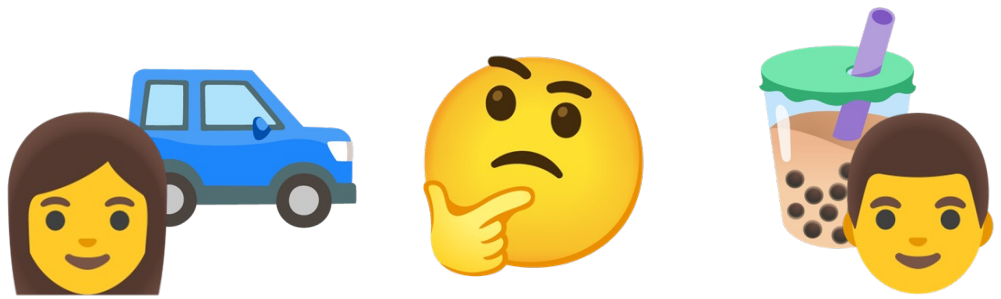
Single log: $tx_1, tx_2, tx_3, \ldots$

Protocol designer needs to choose?

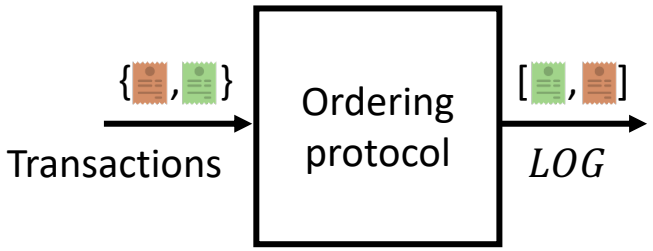
Impossible!

Accountable *prefix* log | Available *full* log





Dynamic Availability + Accountable Safety

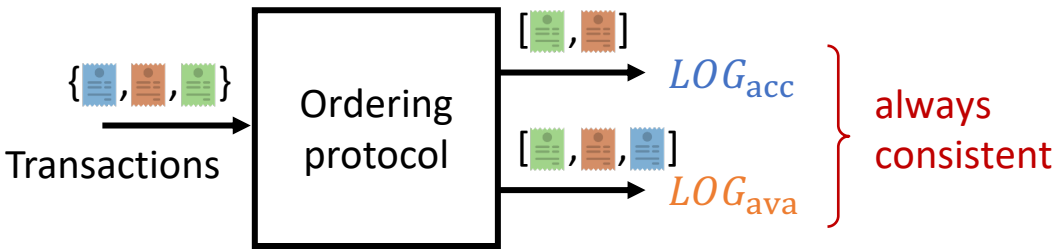


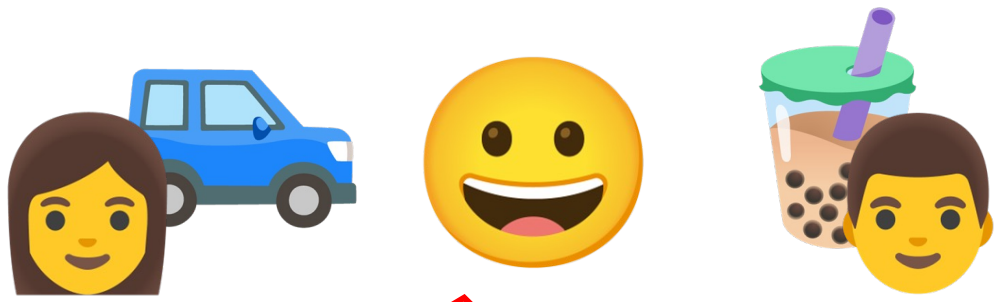
Single log: $tx_1, tx_2, tx_3, \ldots$

~~Protocol designer needs to choose?~~

Impossible!

Accountable *prefix* log | Available *full* log

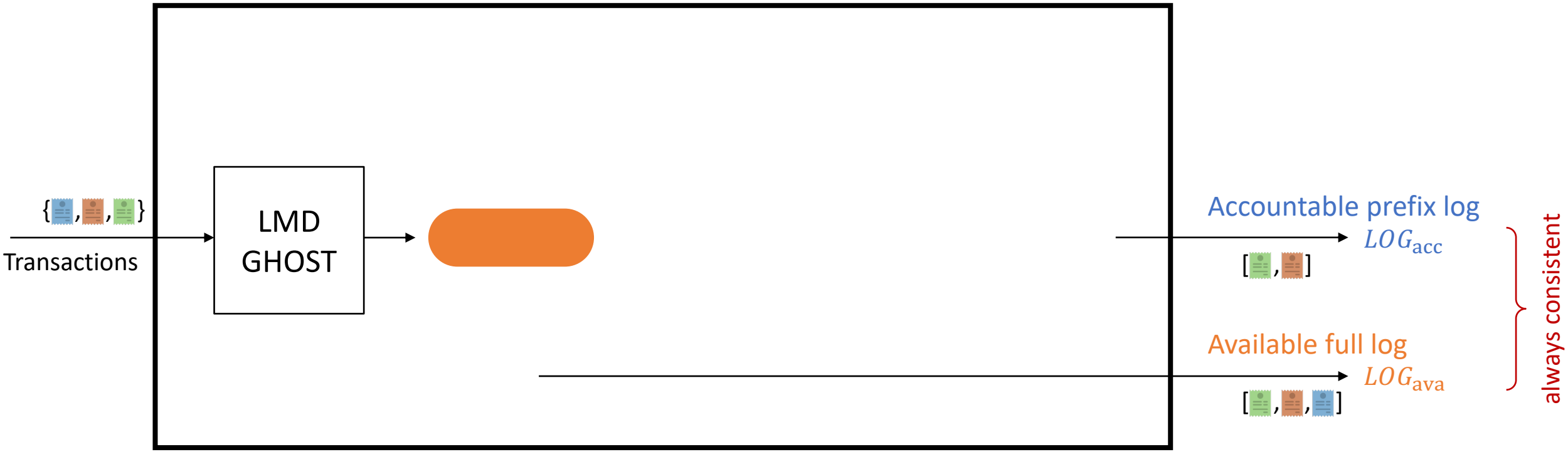




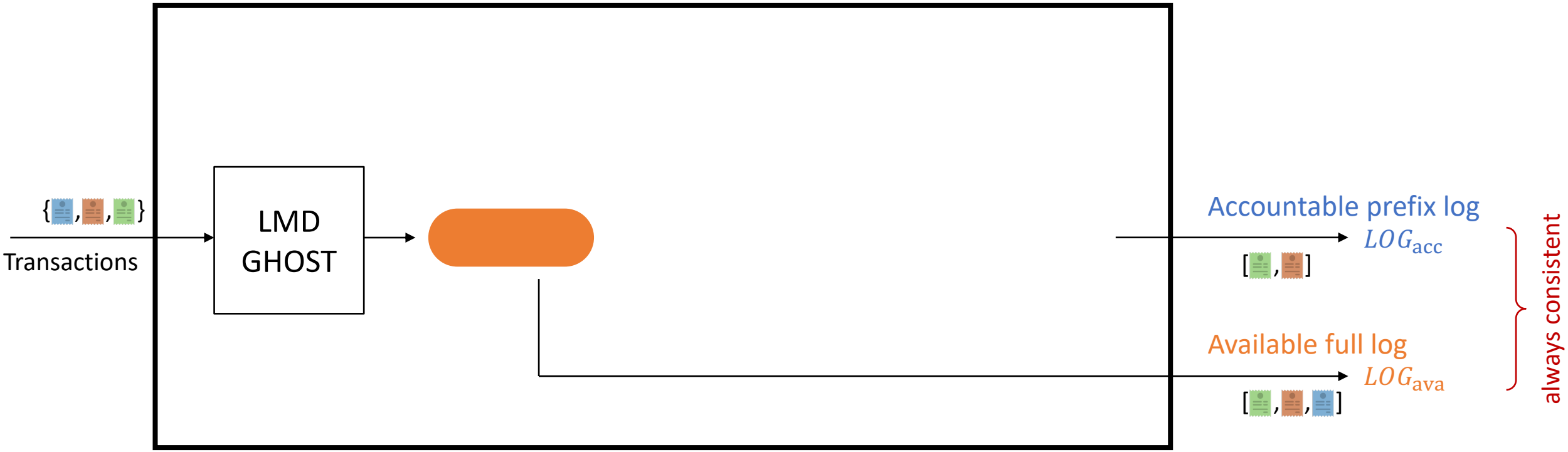
Each user can choose!

Ethereum Consensus

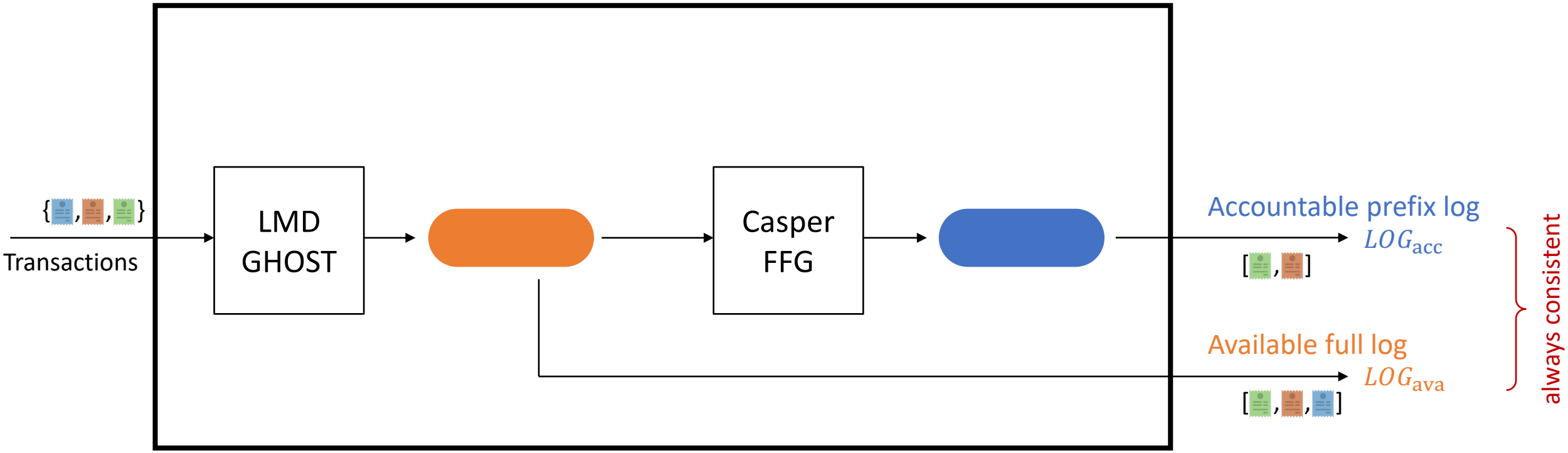
Ethereum Consensus



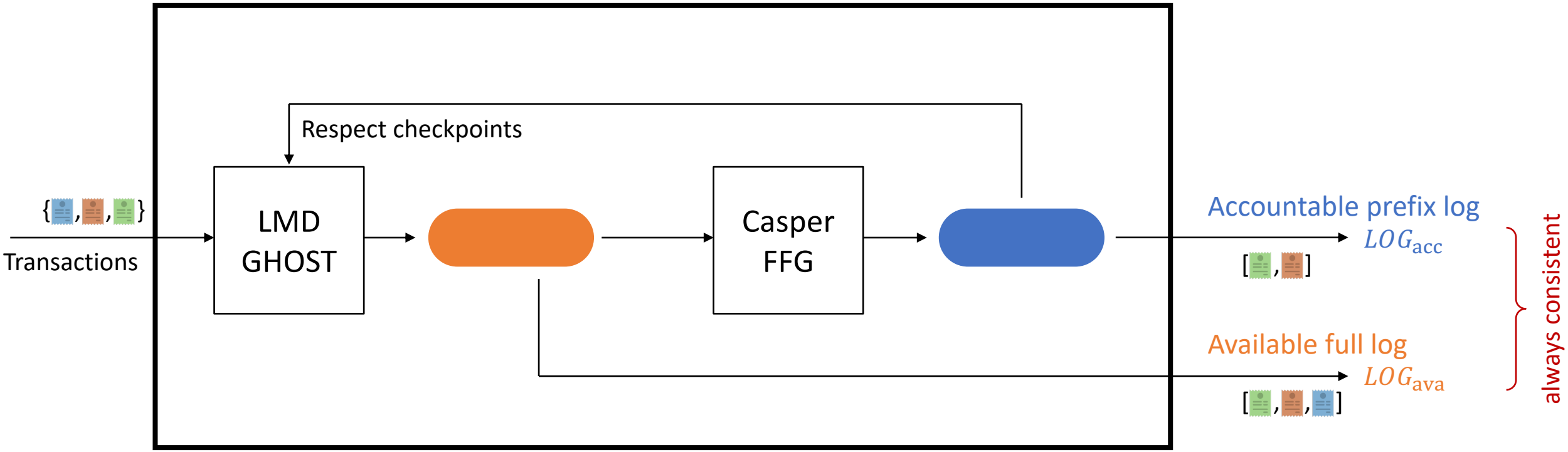
Ethereum Consensus



Ethereum Consensus

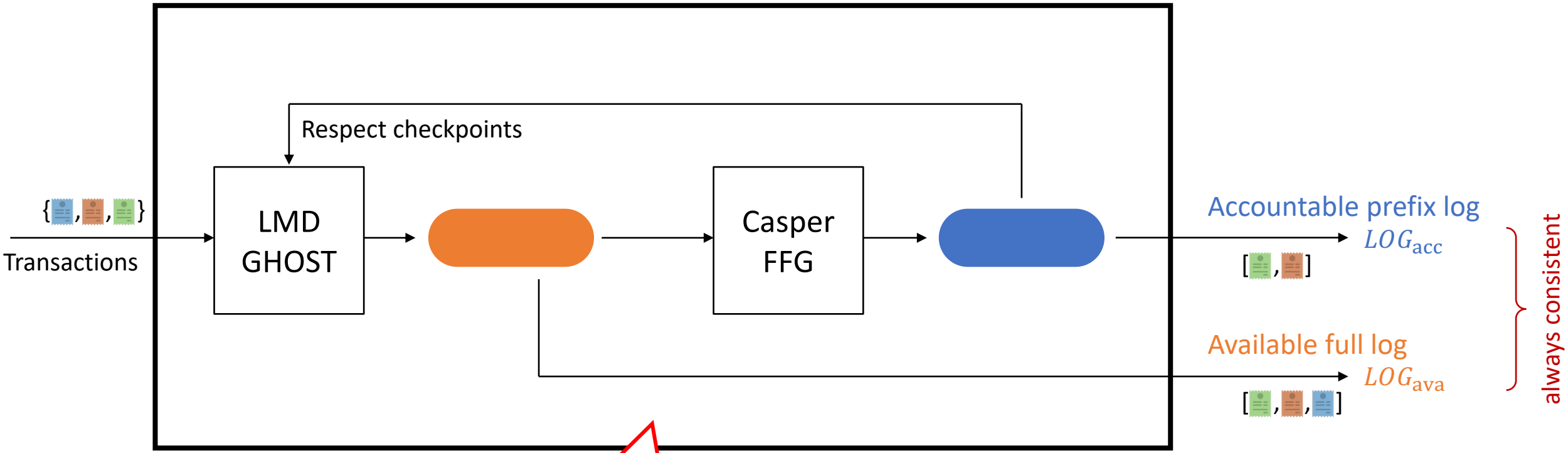


Ethereum Consensus



Ethereum Consensus

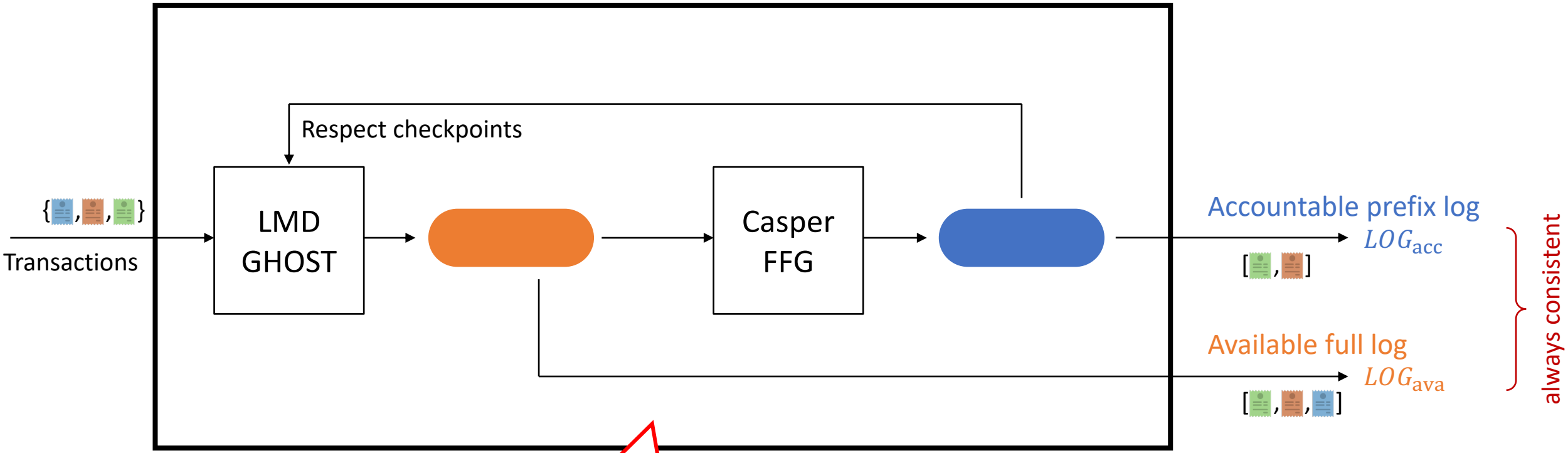
"Combining GHOST and Casper", Buterin et al. 2020
Construction: Neu et al. 2021, Sankagiri et al. 2021, Neu et al. 2022, Neu et al. 2024
Attacks: Schwarz-Schilling et al. 2022, Neu et al. 2022



May look innocuous, but took a series of papers to get right!

Ethereum Consensus

"Combining GHOST and Casper", Buterin et al. 2020
Construction: Neu et al. 2021, Sankagiri et al. 2021, Neu et al. 2022, Neu et al. 2024
Attacks: Schwarz-Schilling et al. 2022, Neu et al. 2022



May look innocuous, but took a series of papers to get right!

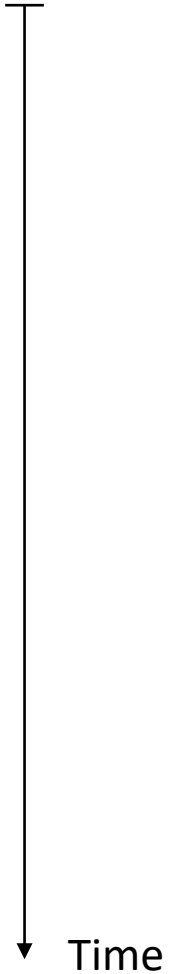
"Don't roll your own crypto(graphy)"
→ "don't roll your own consensus"

Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\mathrm{permissioned}}$

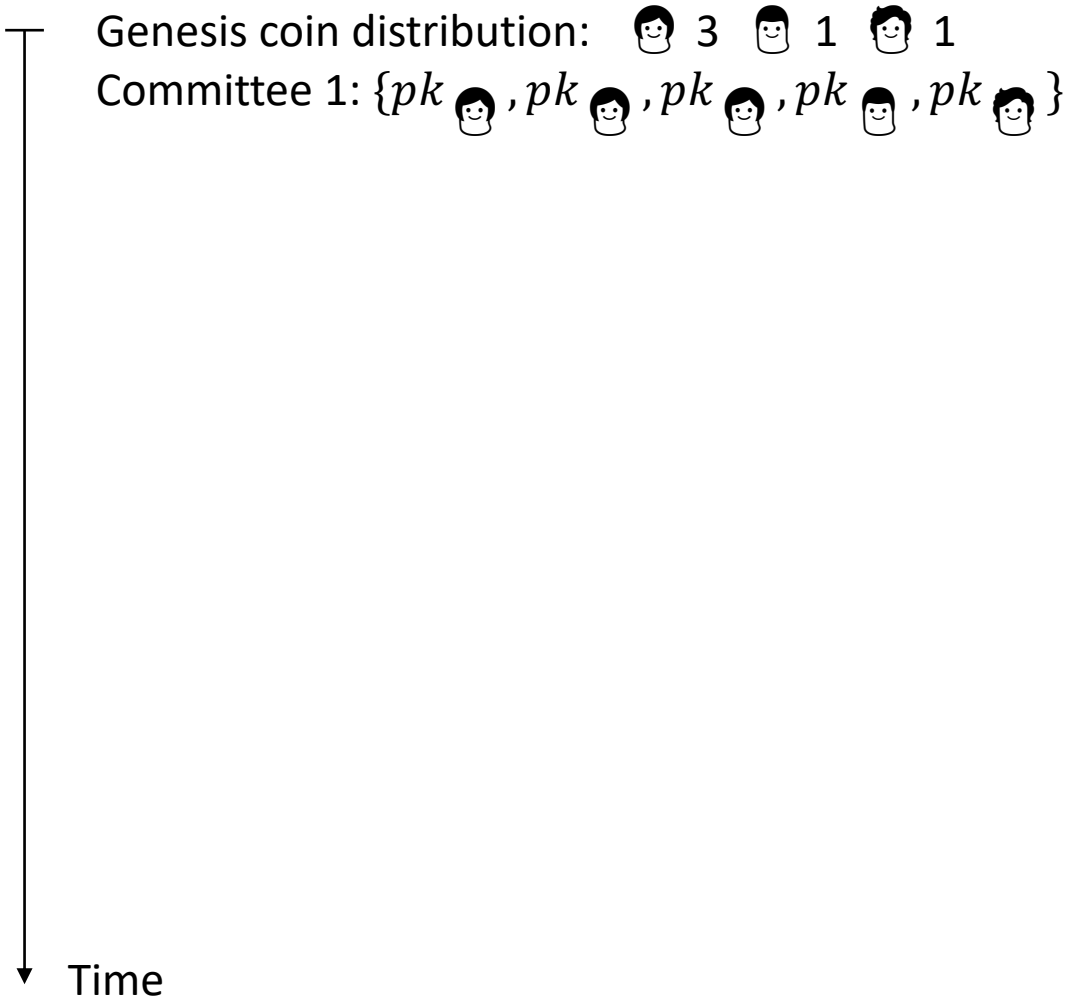
Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\mathrm{permissioned}}$



Proof-of-Stake

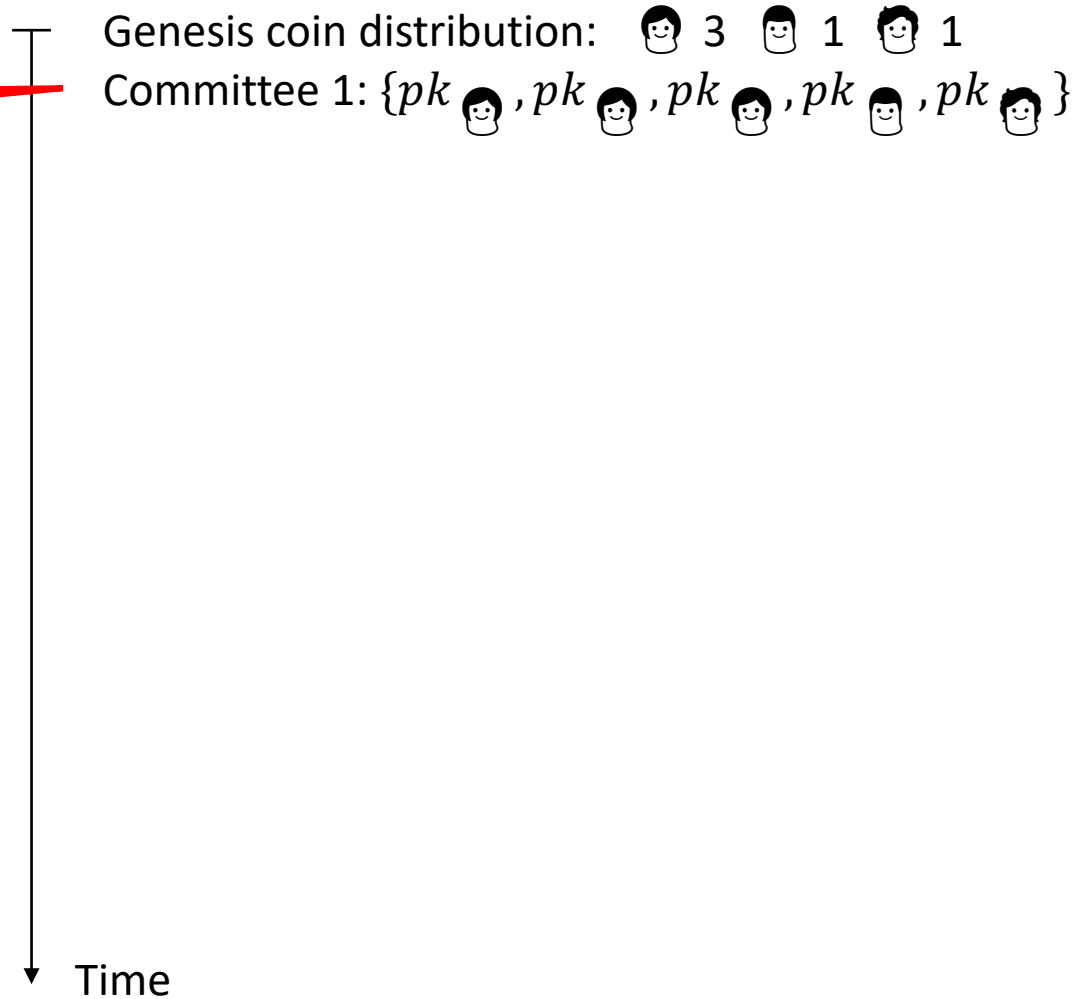
How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

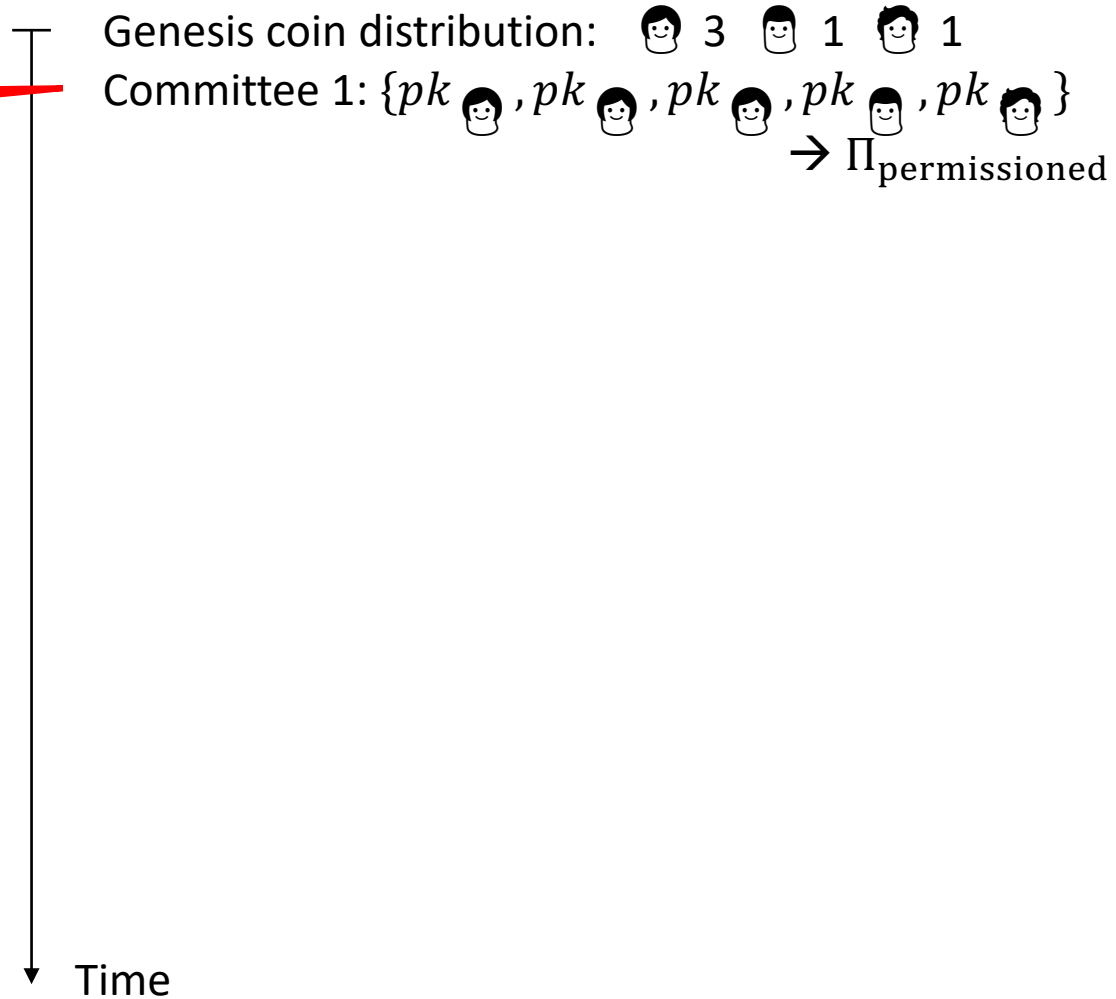
Subsample committee of consensus nodes from coin holders



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

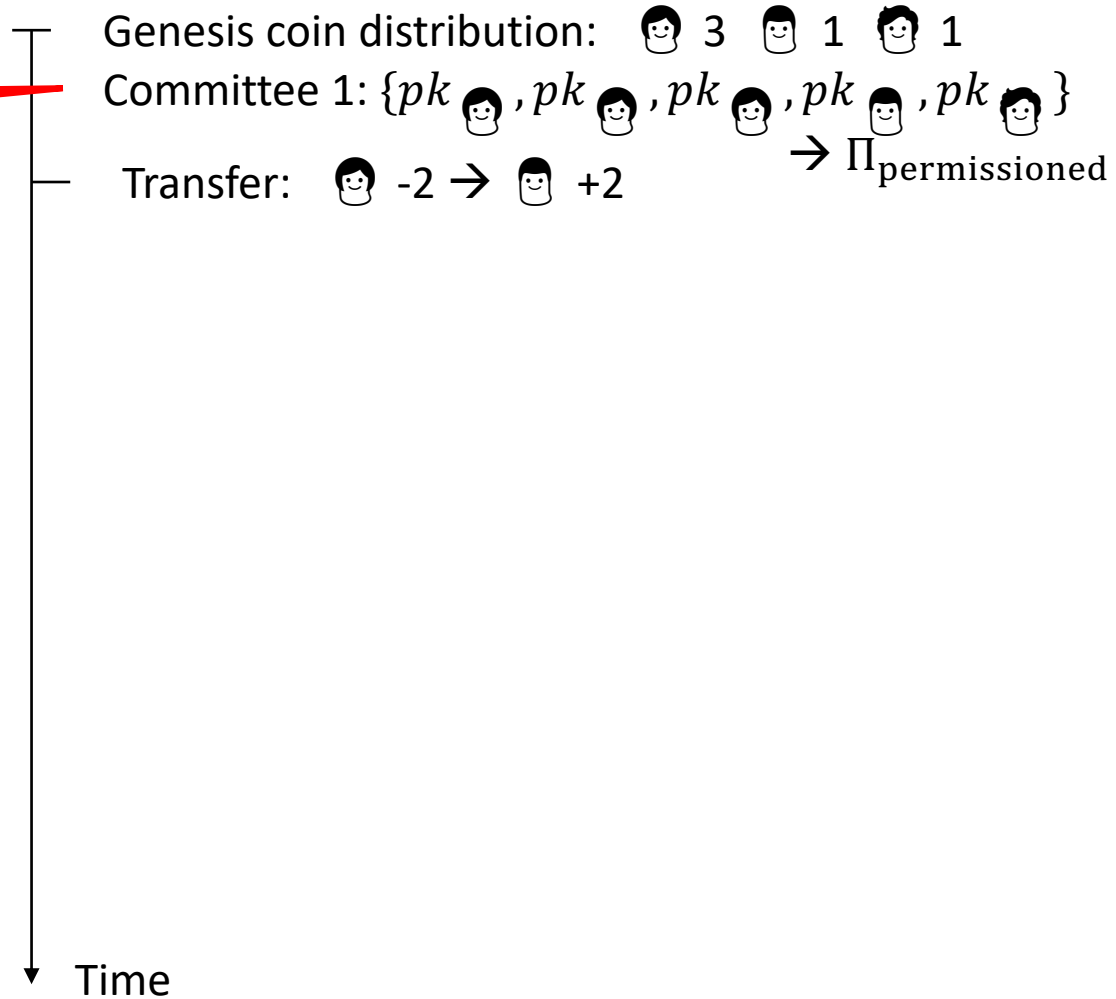
Subsample committee of consensus nodes from coin holders



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

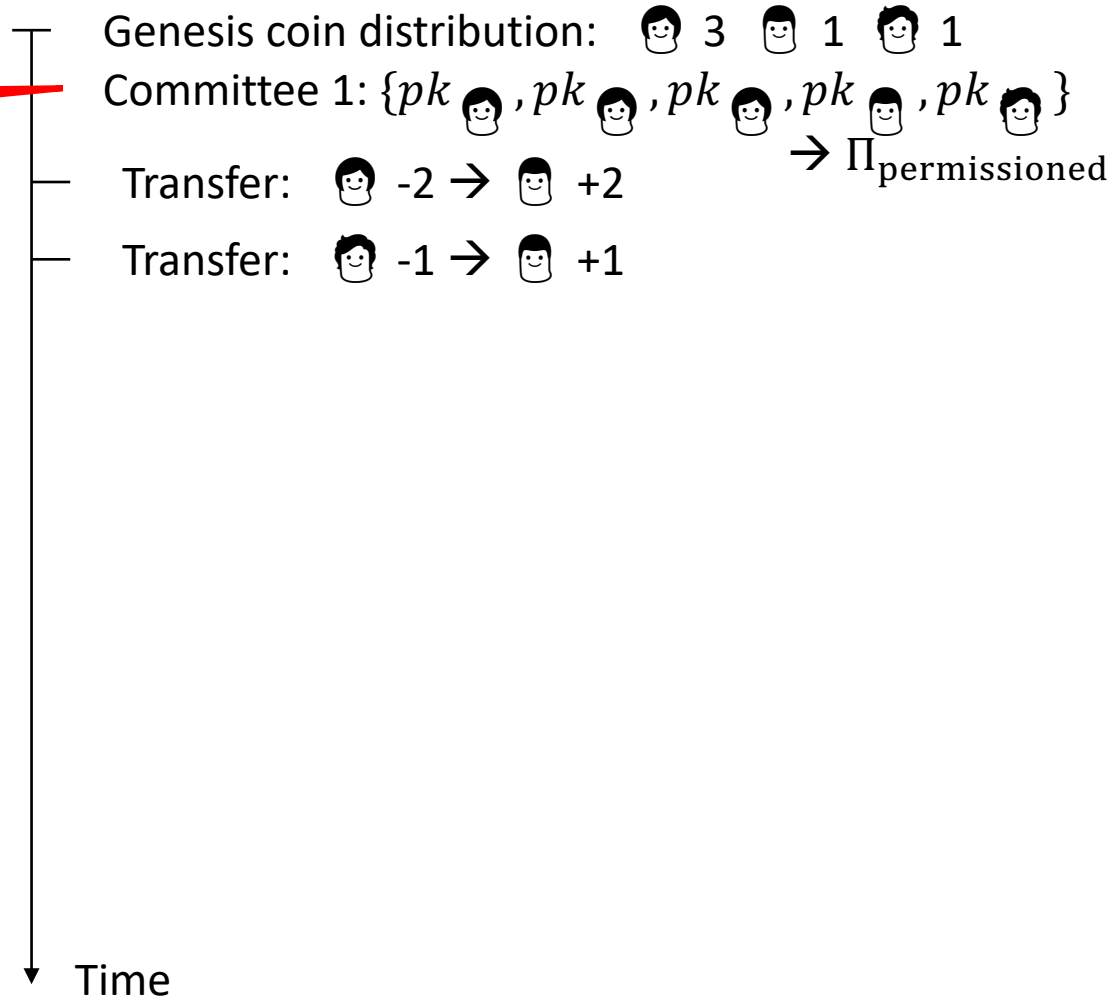
Subsample committee of consensus nodes from coin holders



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

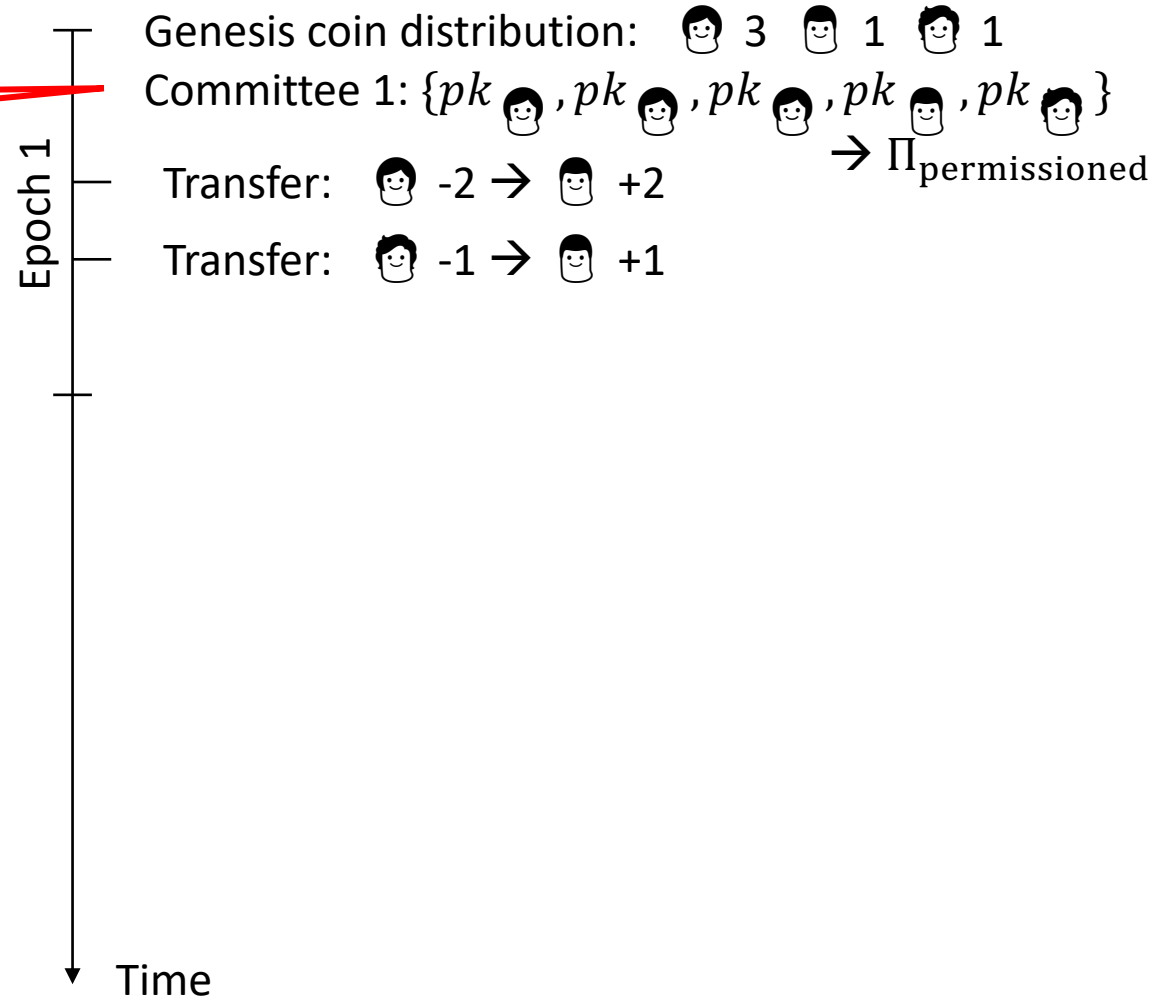
Subsample committee of consensus nodes from coin holders



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

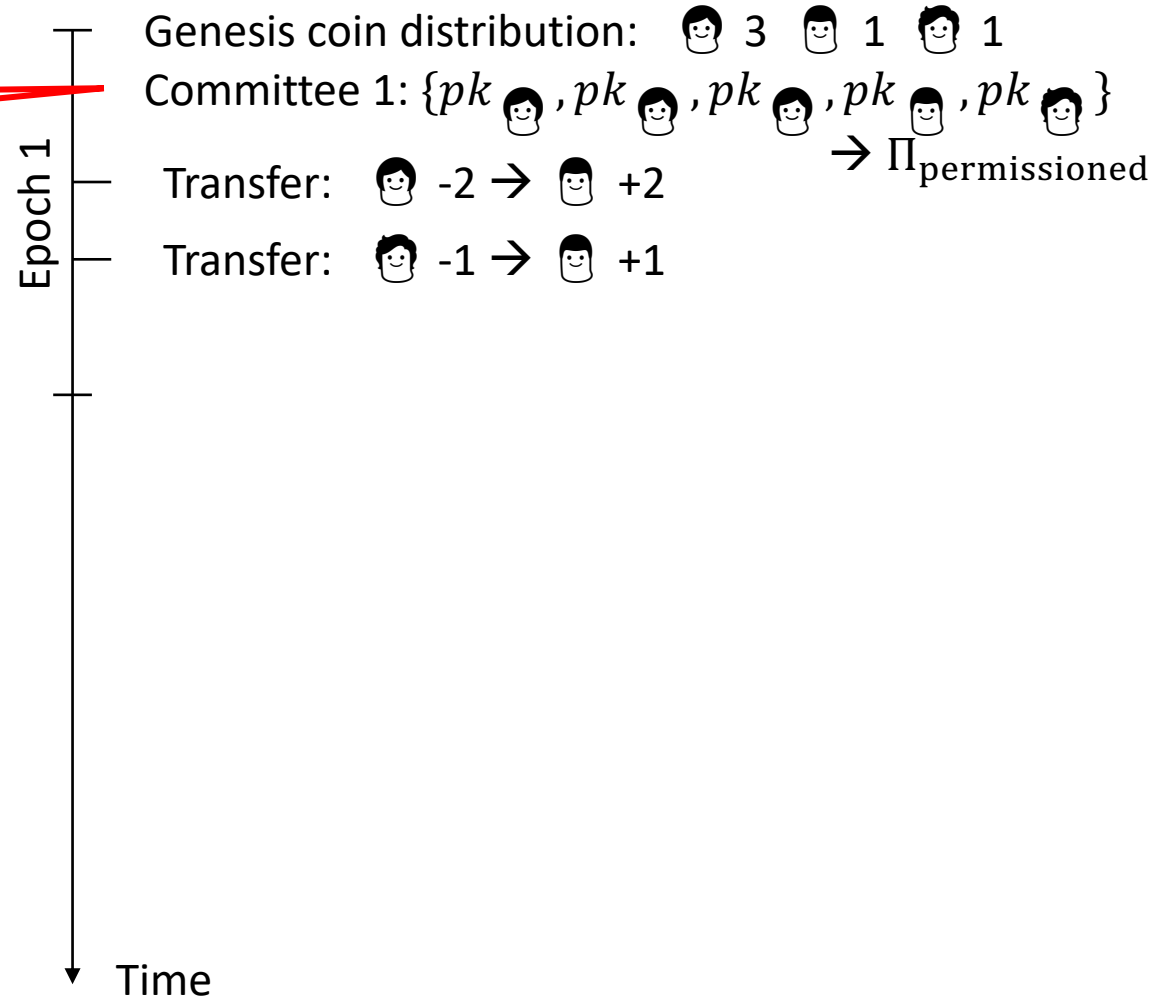


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs

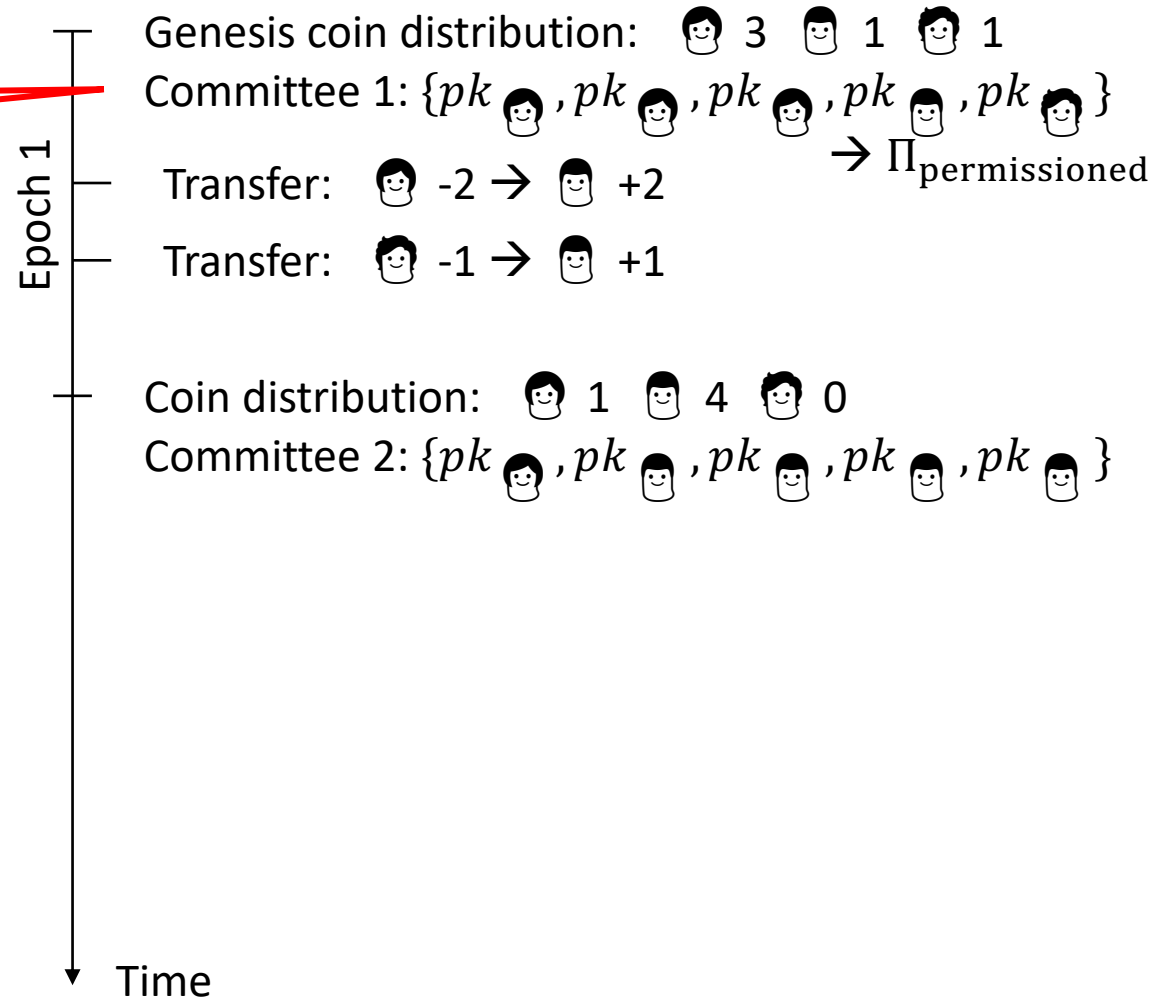


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs

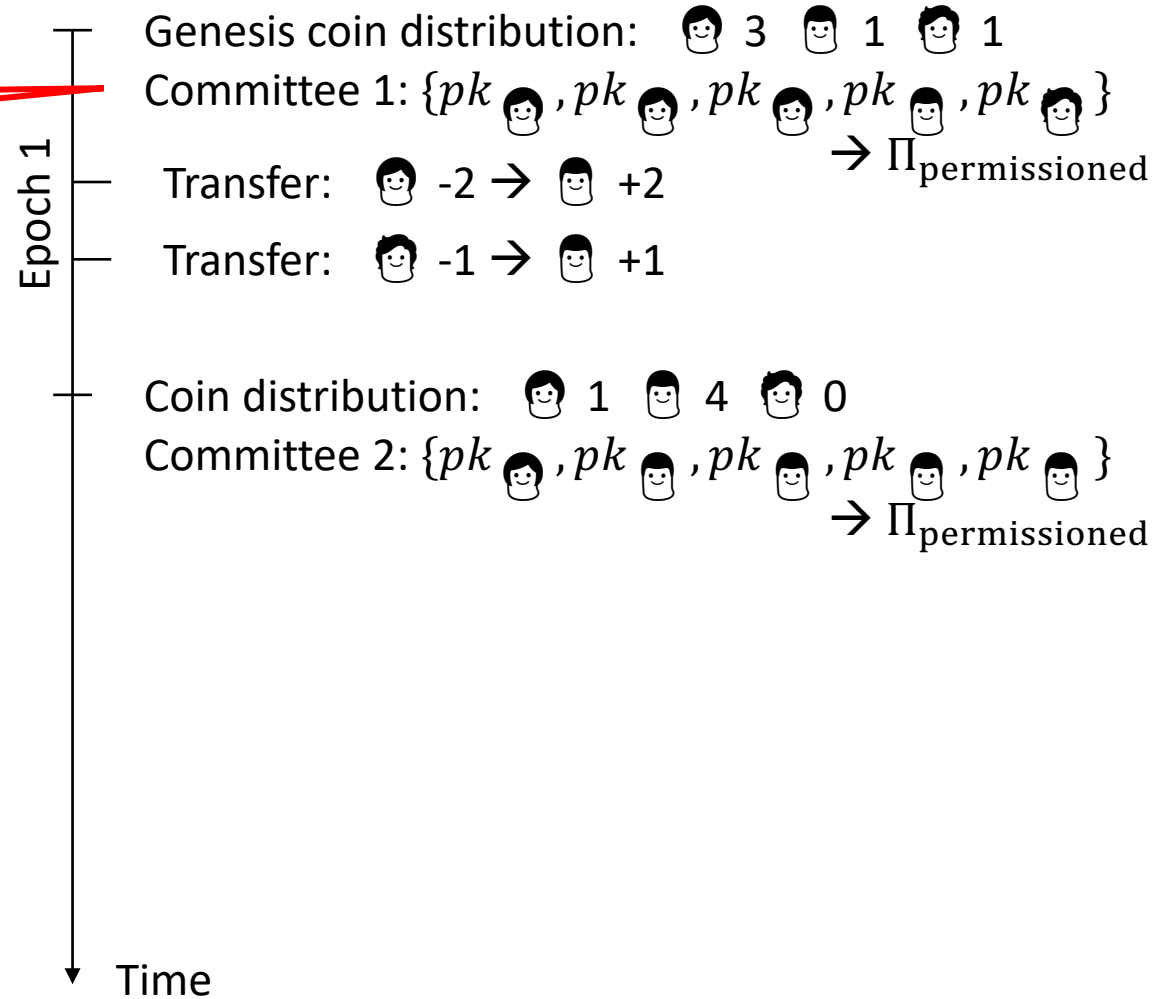


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs

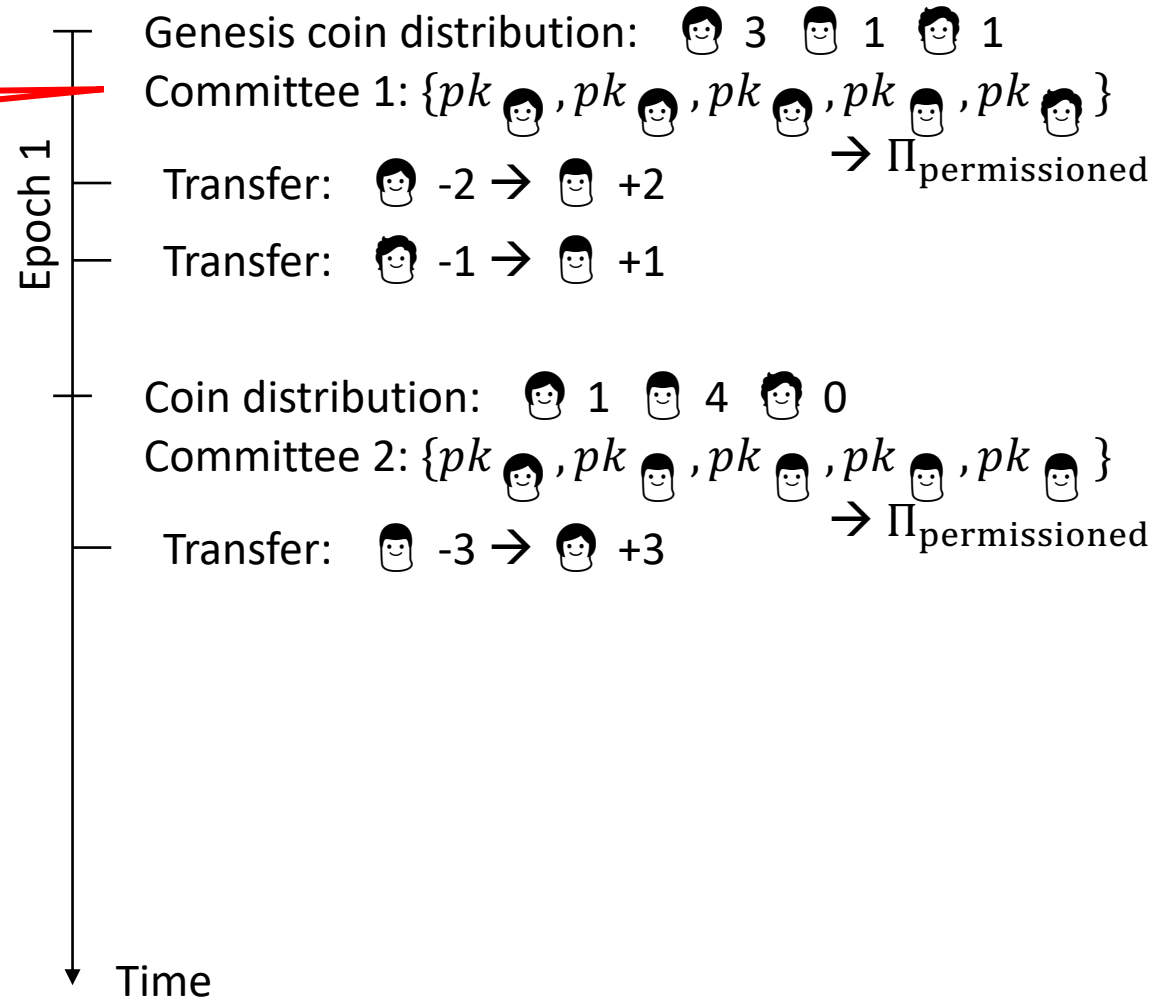


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs

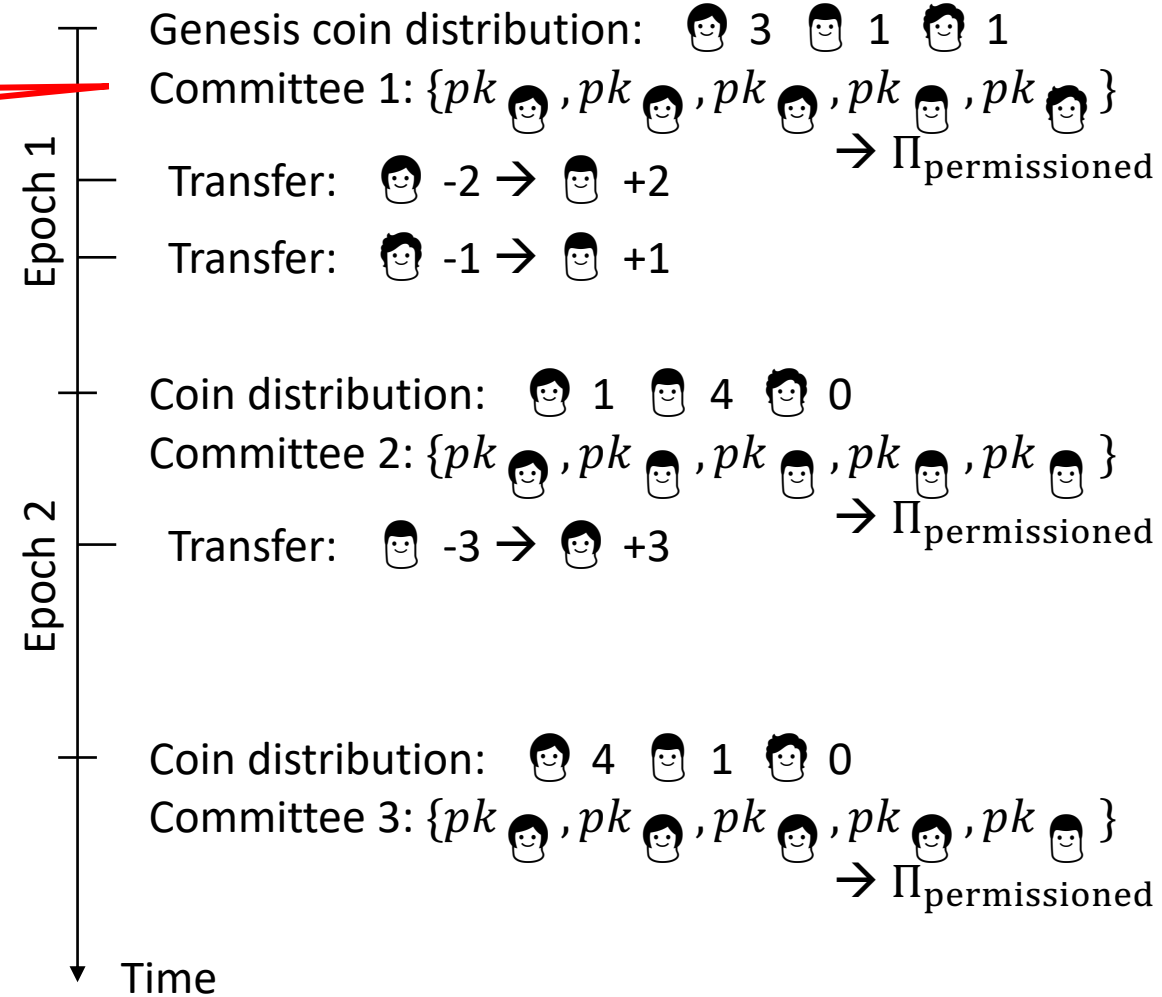


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs

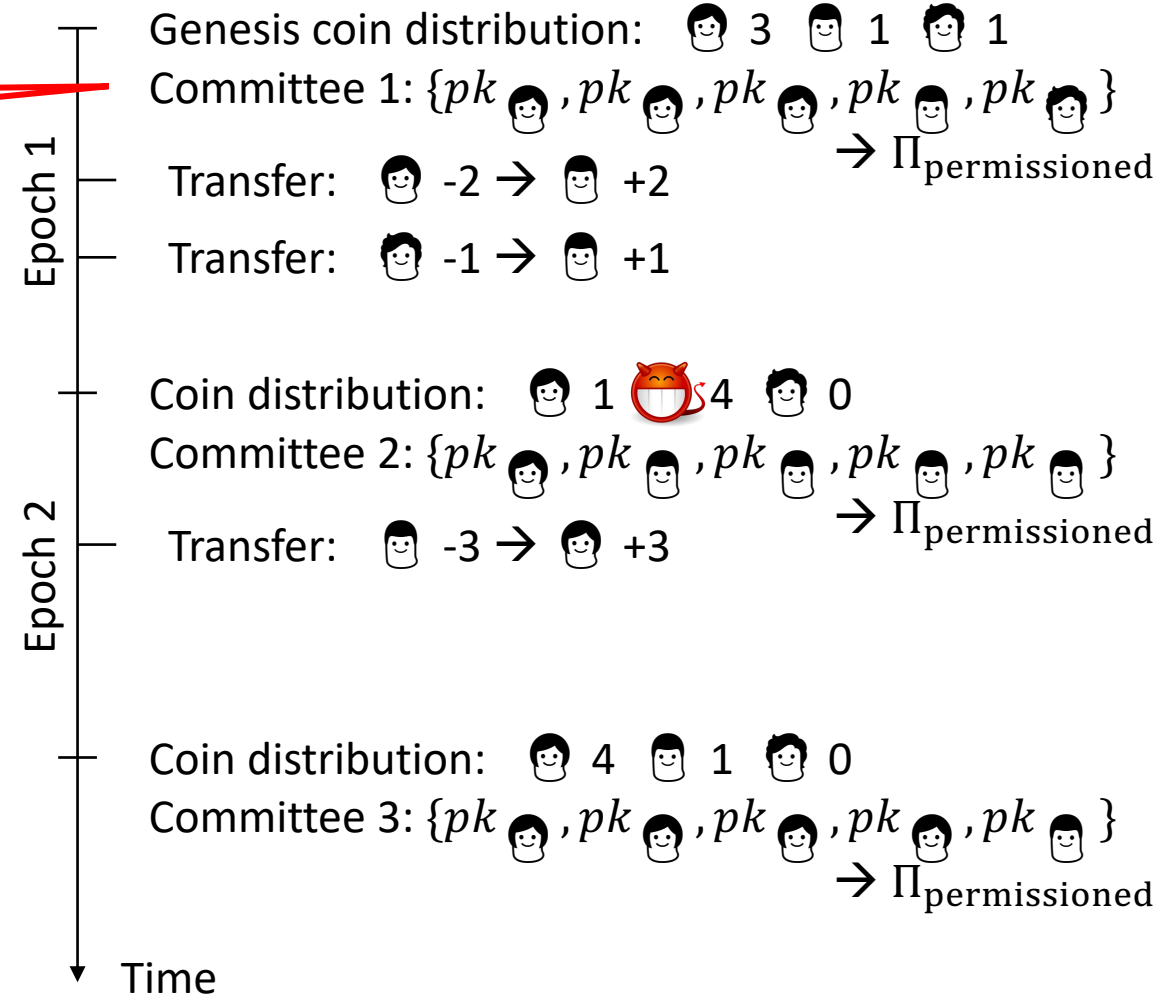


Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs



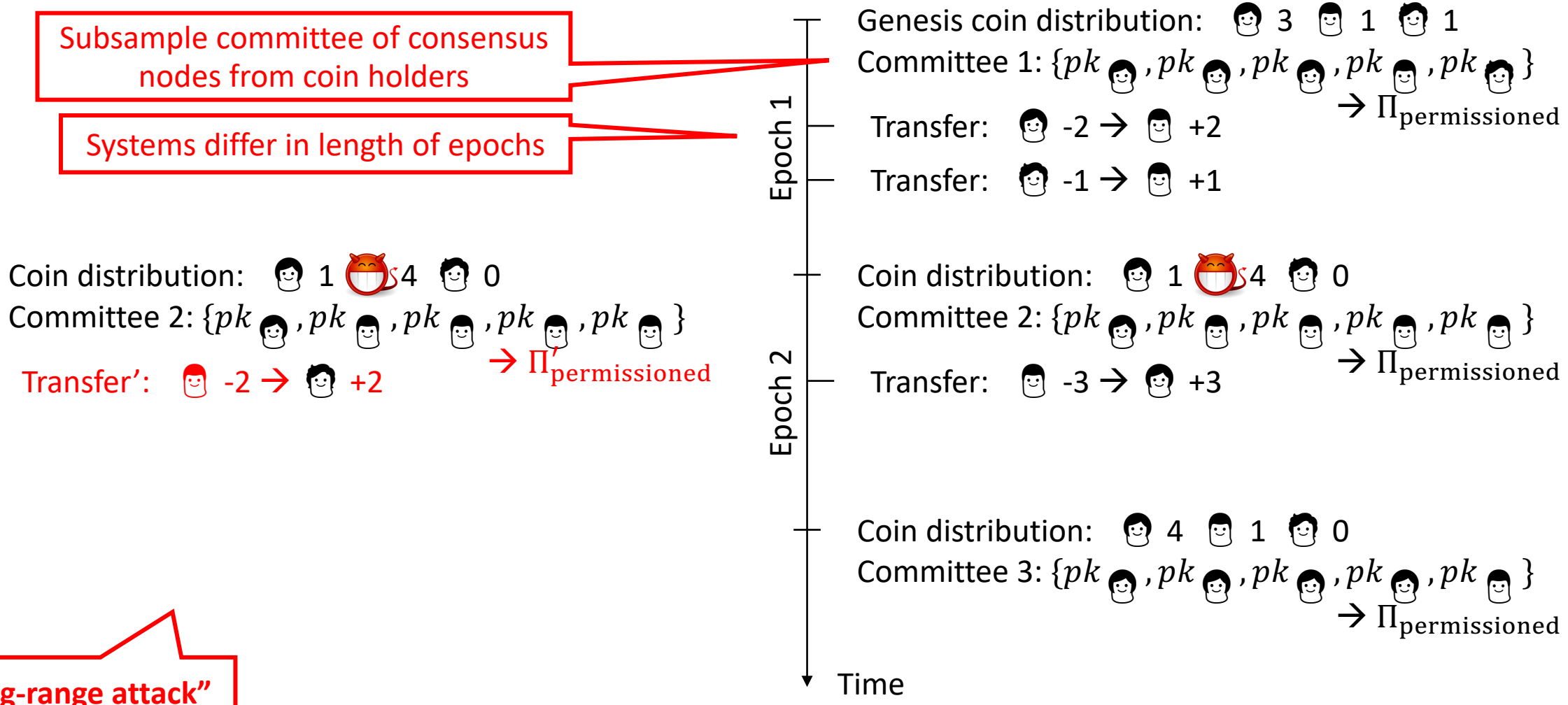
"Long-range attack"

Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$

Subsample committee of consensus nodes from coin holders

Systems differ in length of epochs



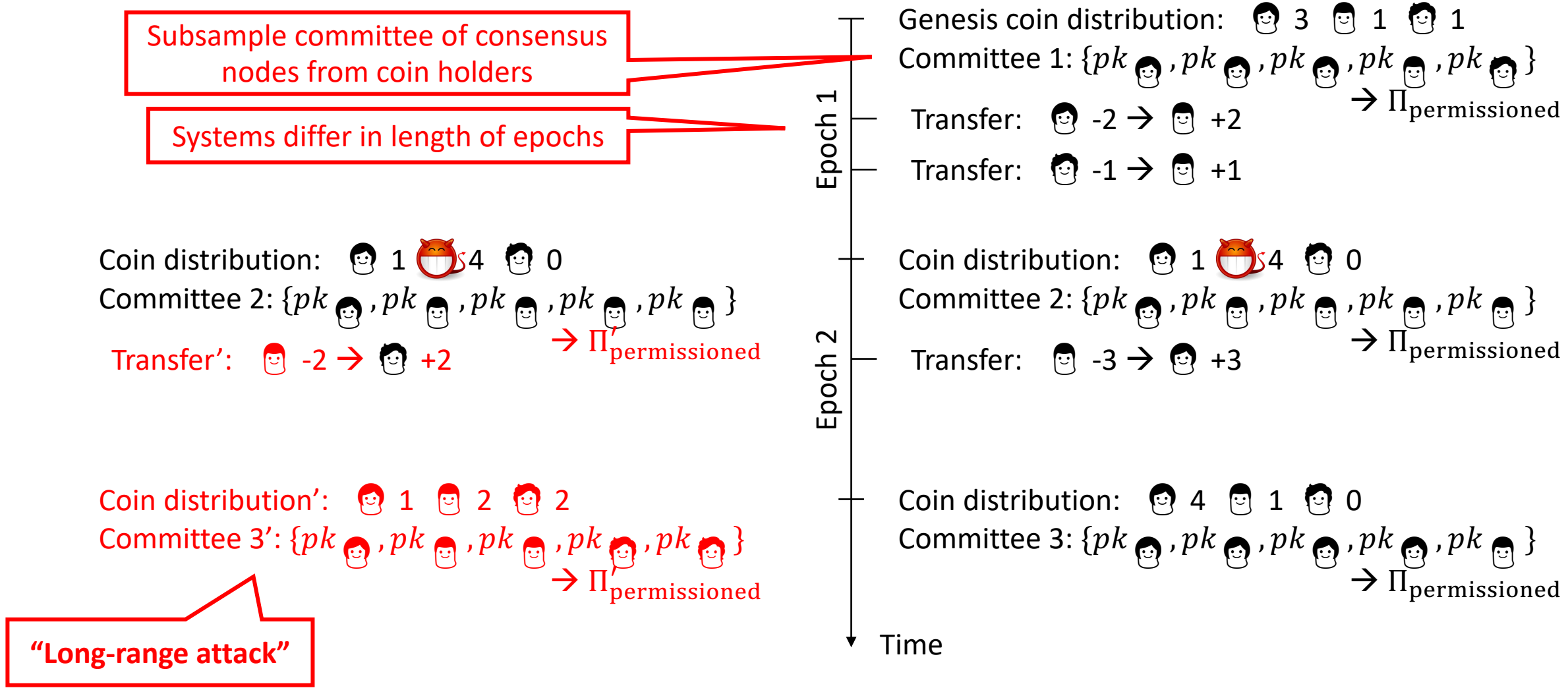
Coin distribution: 1 4 0
Committee 2: $\{pk, pk, pk, pk, pk\}$
→ $\Pi'_{\text{permissioned}}$

Transfer': -2 → +2

"Long-range attack"

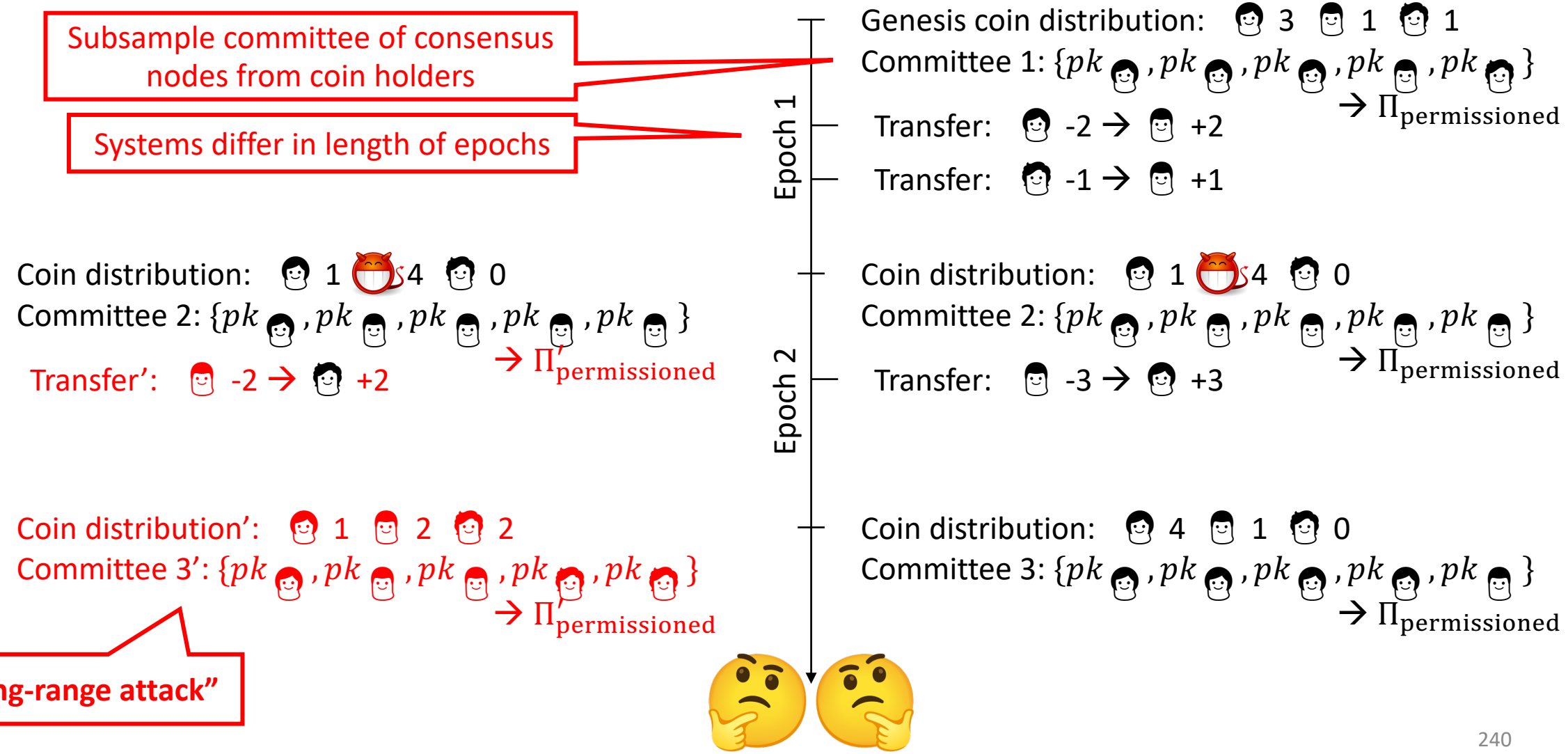
Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$



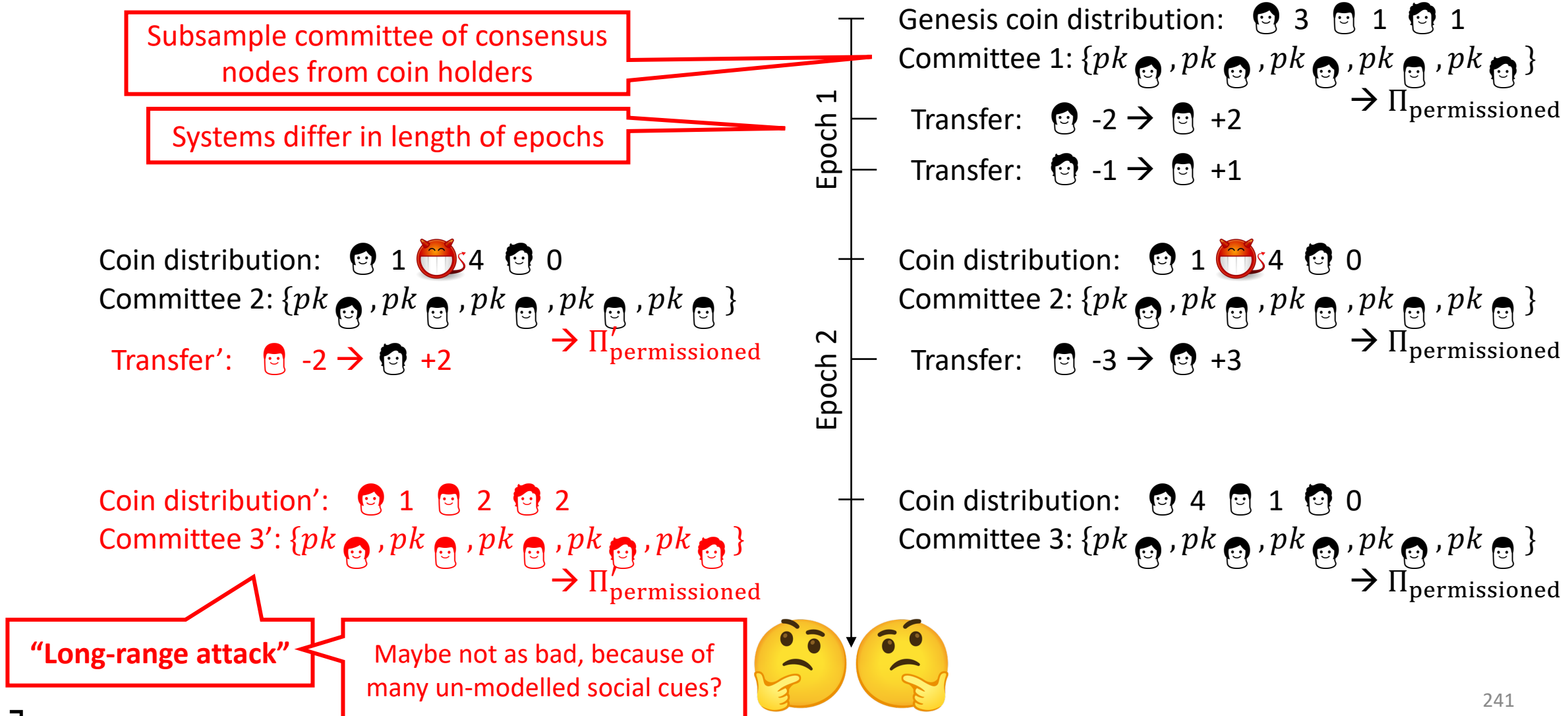
Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$



Proof-of-Stake

How to construct Π_{PoS} from $\Pi_{\text{permissioned}}$



Credits / Attribution

- eCash slides are screenshots from an earlier lecture's slides
- Icons
  - Google Noto Color Emoji
    https://github.com/googlefonts/noto-emoji
  - Twitter Twemoji
    https://github.com/twitter/twemoji
  - Tango icons
    https://commons.wikimedia.org/wiki/Tango_icons
- Logos belong to the respective blockchain projects / companies