Federated Byzantine Consensus

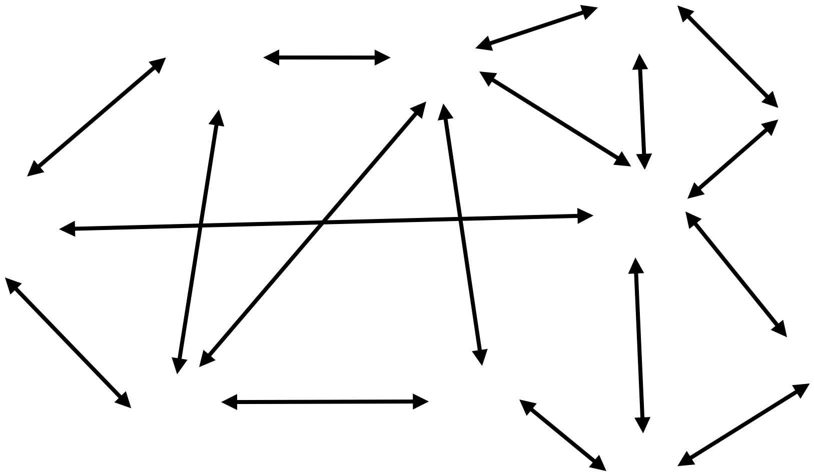
CS839 – Kai Mast

Blockchains so far (simplified)

Peer-to-Peer

(Public, Permissionless, …)

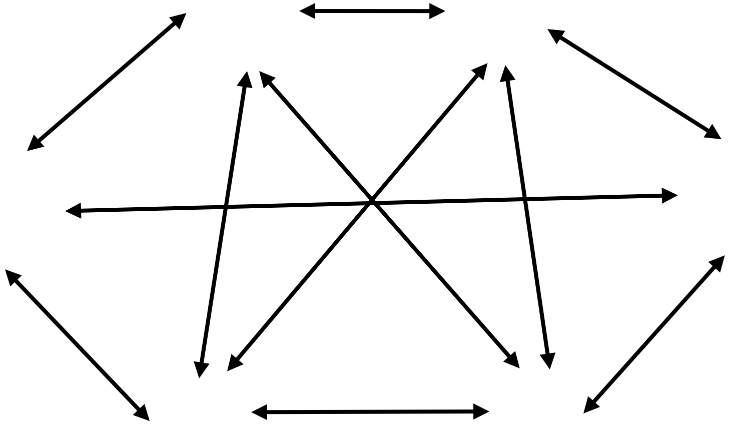
Voting power is function of stake, processing power, etc.



Fully-Connected

(Private, Permissioned, …)

Everyone has one vote

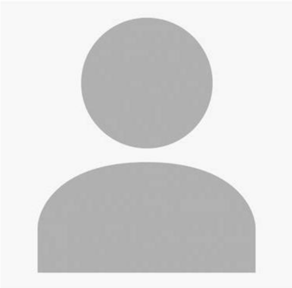


Stellar

- Core idea: Everyone trusts a different set of organizations
- Blockchain network online since 2014
- Stellar mostly serves as a decentralized exchange today
- Around $6.6 billion in market capitalization (as of Dec '21)
- Published in 2019 at SOSP
  - Authors include Marta, Lokhava, David Mazières, Jed McCaleb, and Graydon Hoare

Marta Lokhava
(Stellar Development Foundation)

Jed McCaleb
(Stellar Development Foundation)



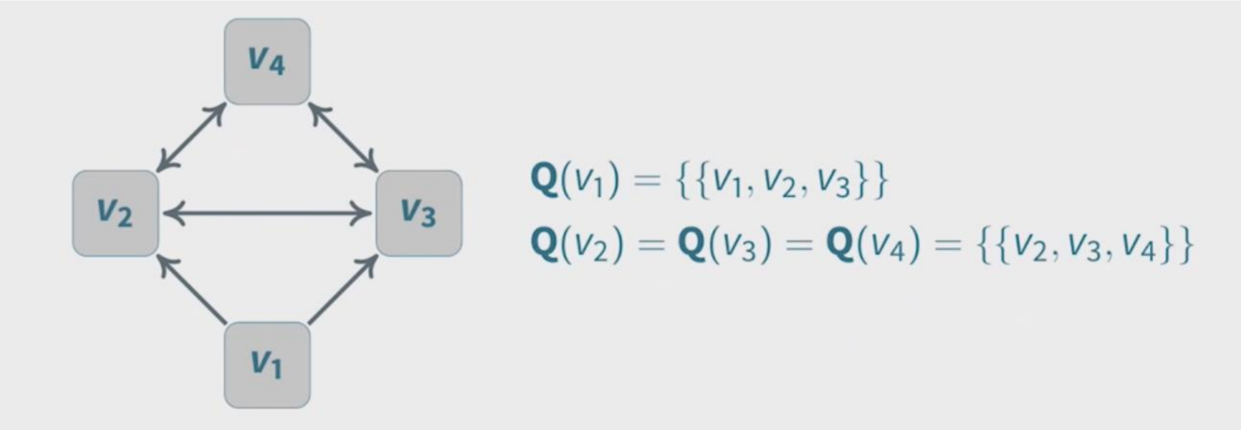
Graydon Hoare
(Stellar Development Foundation)

David Mazières
(Stanford /
Stellar Development Foundation)

Quorum Slices

- Each node v picks a set "slices", if *one* of their slices accepts a transaction, v assumes the entire network has accepted the transaction.
- For example, in PBFT for each node $n \in N$ its quorum slices are all sets $N' \subseteq N$ where $|N'| > 2f$.
- Quroum slices can express different levels of trust
  - e.g., a node could have one quorum slice consisting of just one other node, and another quorum slice consisting of 10 nodes

Quorums in Stellar



$$\mathbf{Q}(v_1) = \{\{v_1, v_2, v_3\}\}$$
$$\mathbf{Q}(v_2) = \mathbf{Q}(v_3) = \mathbf{Q}(v_4) = \{\{v_2, v_3, v_4\}\}$$

(Taken from David Mazières' Slides)

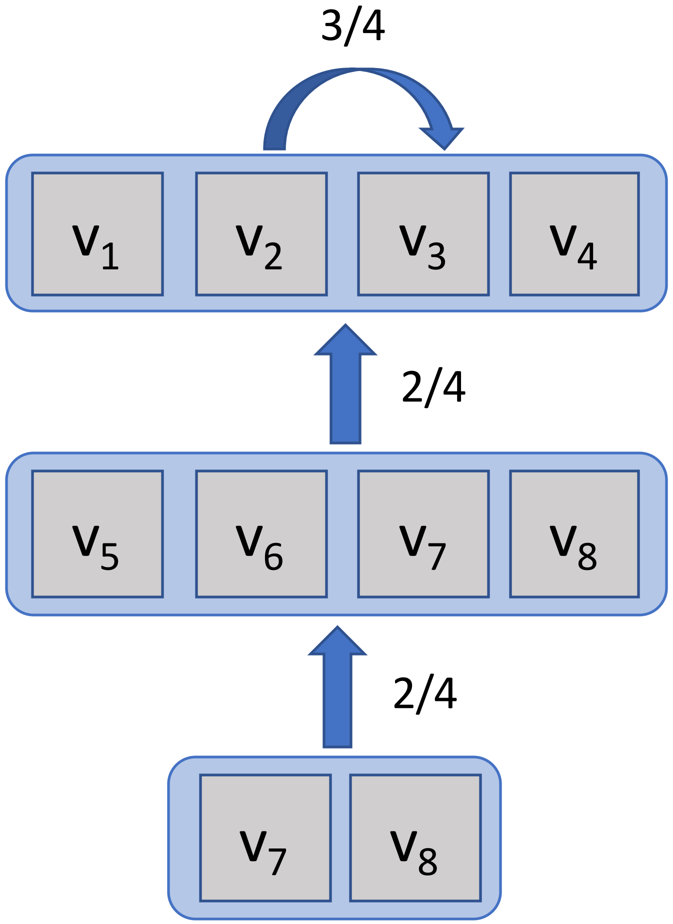
- A quorum contains at least one quorum slice for each node.
- We might need many quorum slices to achieve consensus, but one quorum is sufficient.
- For example, a slice for v_1 is $\{v_1,v_2,v_3\}$, but it's not a quorum because it does not contain v_4

The Internet Hypothesis

"Everyone wants to talk to everyone"

- We say two nodes are intertwined if all combinations of their quorum slices contain at least once common node
- Being intertwined is transitive: if nodes n_1 and $n_{2,}$ and nodes n_2 and n_3 are intertwined, nodes n_1 and n_3 are intertwined as well.
- Internet hypothesis states that all nodes (we care about) are intertwined

Tiered Consensus



Example topology with 3 tiers

Assumption: Like with the Internet there will be a few "top tier" organization, e.g., Google or ISPs

- There's no central organization that defines the topology or who the top-tier nodes are
- Instead, the topology is driven by market forces
- Once top tier agrees, agreement will "trickle down"
- Like in Bitcoin, topology can change at any time and nodes might not know the full topology

Federated Byzantine Consensus

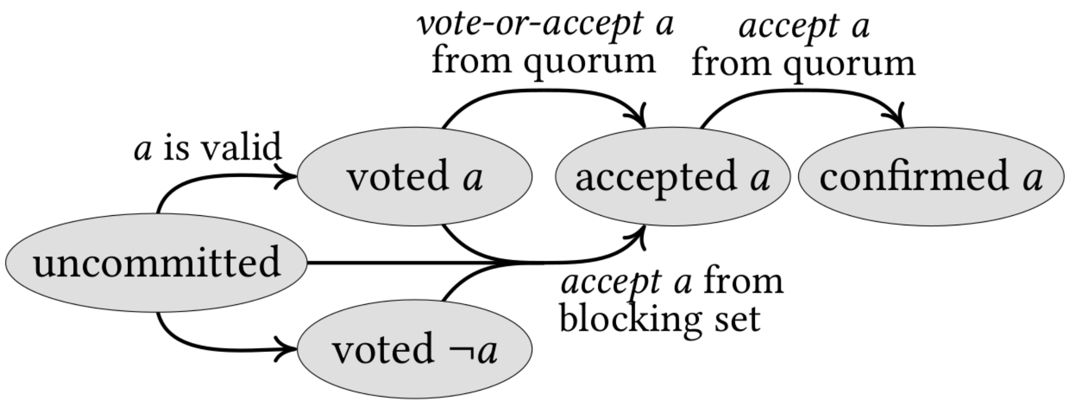
For any set of transactions x, the network can agree on three statements

NOMINATE(x) *"x is a valid decision candidate"*
x does not conflict with the current state

PREPARE(x, n) *"No value other than x was or will ever be decided in any ballot ≤ n."*
(The round might still fail, and no agreement could be made)

COMMIT(x, n) "x is decided in ballot n"

Voting on Statements



Stages of Federated Voting

Each node votes at most once to either for or against a statement

- They cannot change their vote for the same statement

Once a node sees a quorum slice vot for the statement they vote to accept the statement

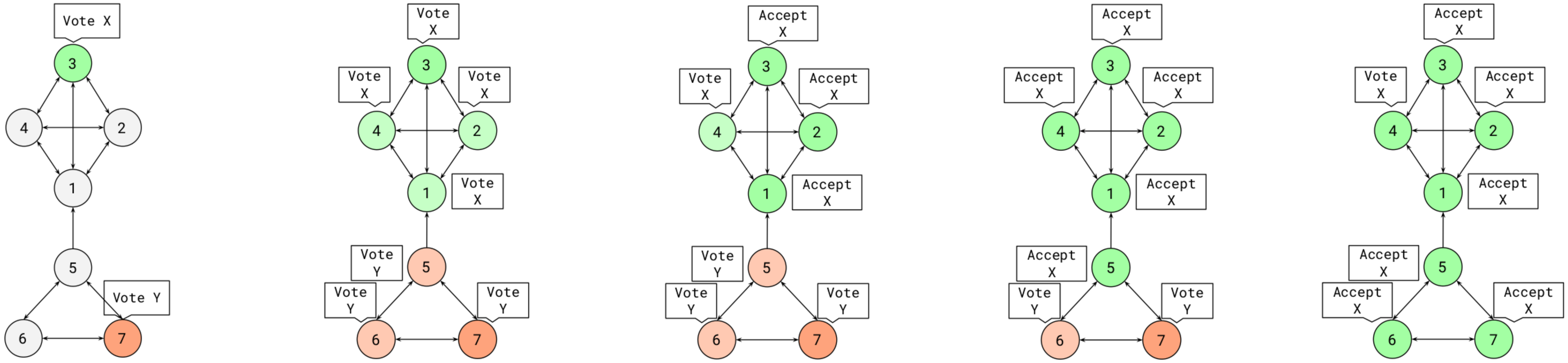
- Nodes can be "overruled"; correct nodes will accept statements they voted against

Blocking Sets

A blocking set B_v is a set of nodes that intersects every slice of a node v

- v will not accept a statement that B_v voted against
- v will not be able to make progress if its B_v is faulty

Blocking Sets: Example



(a)

Nodes 3 and 7 vote for contradictory statements

(b)

Nodes vote for statements (both are valid)

(c)

Nodes 1,2, and 3 vote to accept, because they saw a valid quorum slice

(d)

The blocking set for node 5 is node 1. Node 5 votes to accept X instead of Y.

(e)

Eventually the entire network votes to accept X

Nomination Process

Problem: There might be too many potential set of transactions to vote on.

Idea: Nominate potential transaction sets before voting on a round

- Nodes can vote for multiple, non-contradictory NOMINATE statements
  - Correct nodes will vote for the union of the transaction sets they have already seen
- Once they, see a quorum for a NOMINATE statement, they stop voting and accept the statement
- There can be multiple accepted NOMINATE statements per round, but only a finite number

Leaders in Stellar

Idea: Reduce the set of nominated transaction sets by having only leaders propose them.

- We cannot pick a single leader like in PBFT (the set of members is variable and potentially unknown)
- Instead, leaders are picked using a random function
- There might be multiple leaders at the same time

Nodes pick the node(s) with the highest priority as their leader, where priority is defined as:

$H(m) = \text{SHA256}(b \,\|\, r \,\|\, m)$

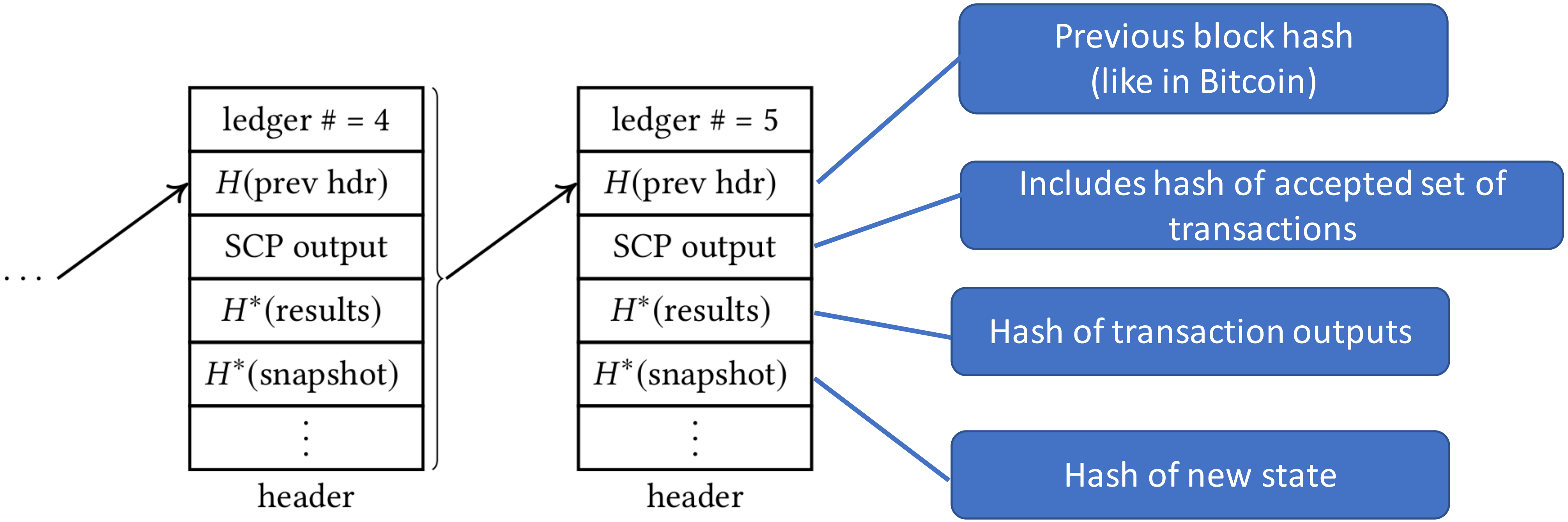
- b is the current ballot number
- r is the leader election round (we move to the next round only if a leader fails)
- m is the node identifier
- Hash value is weighted by slice size (not shown here)

Round (or Ballot) Synchronization

Stellar has a timeout mechanism that decides when to "give up" and move to the next round

- Timer does not start until nodes are part of a PREPARE quorum that is the current or later ballot
    - Ensures all nodes start the timer at roughly the same time
- If we have not prepared a set of transactions yet, we can keep waiting without being stuck
- Like in PBFT, timeouts increase exponentially after each failure

The Stellar Blockchain



Previous block hash (like in Bitcoin)

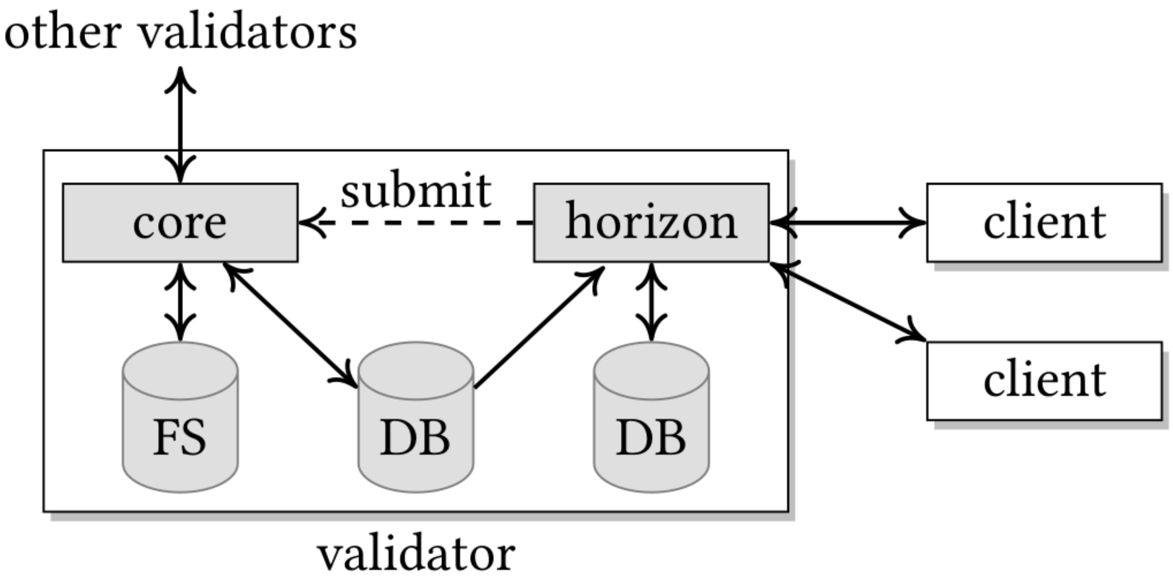
Includes hash of accepted set of transactions

Hash of transaction outputs

Hash of new state

- Stellar agrees on a sequence of blocks
- Once a block is accepted it is finalized and immutable
- The ledger uses an accounts model (like Ethereum)

Stellar Implementation



Stellar-core

- Low-level ledger functionality, e.g., storing transactions
- About 92k lines of C++ code

Horizon

- High-level abstractions, such as accounts and payments
- About 18k lines of Go Code

Bridge (client)

- Integration with existing services, e.g., some user-facing frontend

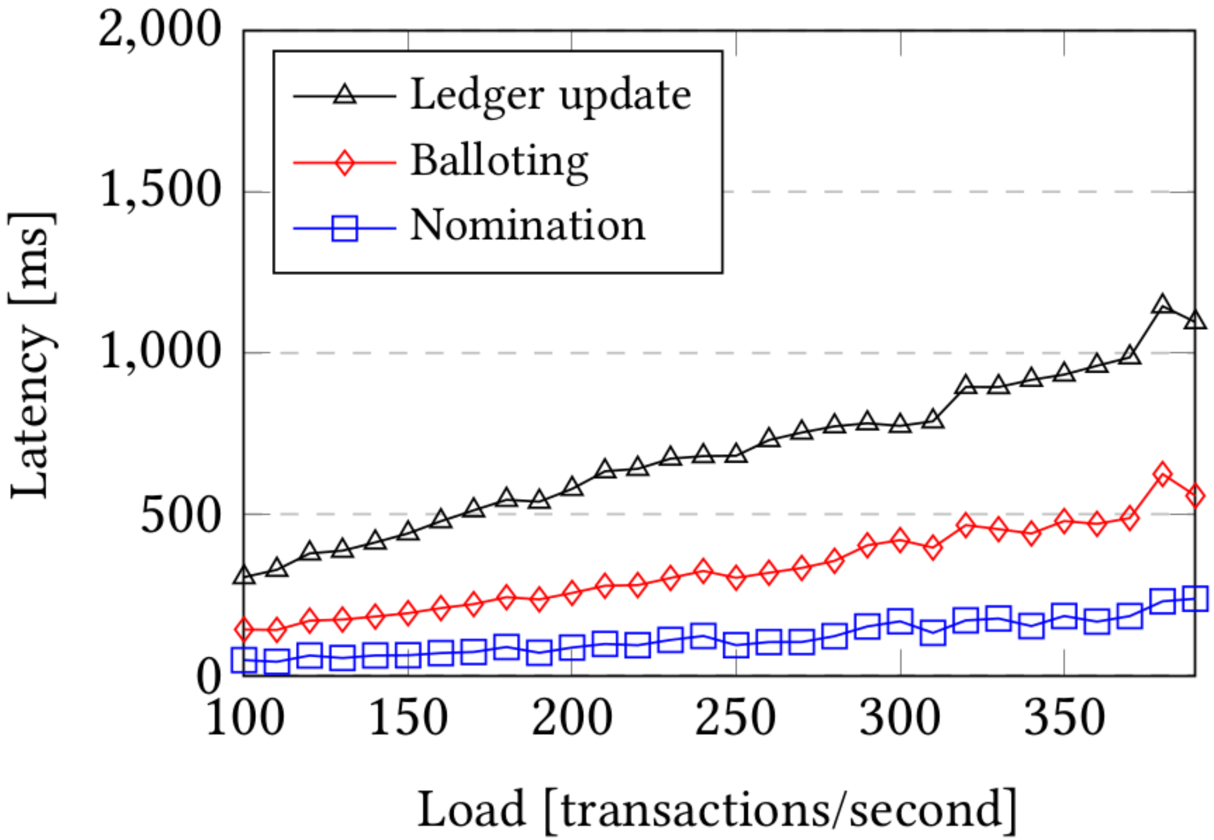
Federation server (client)

- Human-readable naming service for accounts

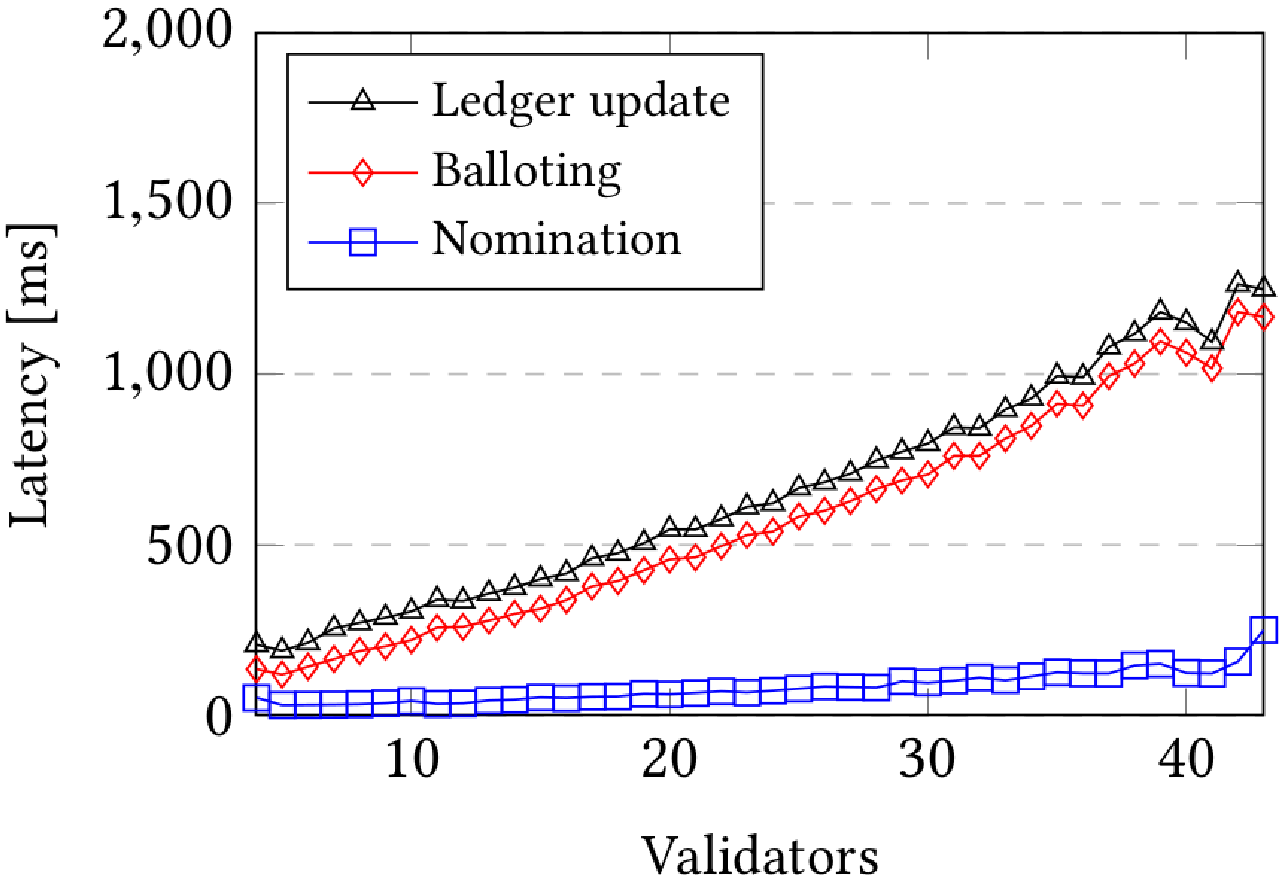
Configuration Complexity

- Stellar needs to guarantee that there do not exist multiple disjoint quorums
  - We need to check that all quorum slices are intertwined (NP-hard)
  - The current network is fairly small (only a few dozen nodes), so this is still feasible
- Stellar needs to prevent "risky" configurations that are close to disjoint
  - A small set of nodes could fail and make the network disjoint
  - The implementation has a warning systems built-in to address this
- Currently, the stellar development foundation's nodes still constitute a blocking set for most of the network

Evaluation: Increasing Transaction Load



Evaluation: Increasing Number of Validators



Discussion

- Do you believe the "Internet Hypothesis"?
- Why not just run PBFT among the high-tier nodes?
- Why is there one globally-replicated state across all tiers?

That's all for today

Next week: Guest lecture about the Bitcoin Decentralization Study **(Zoom only!)**