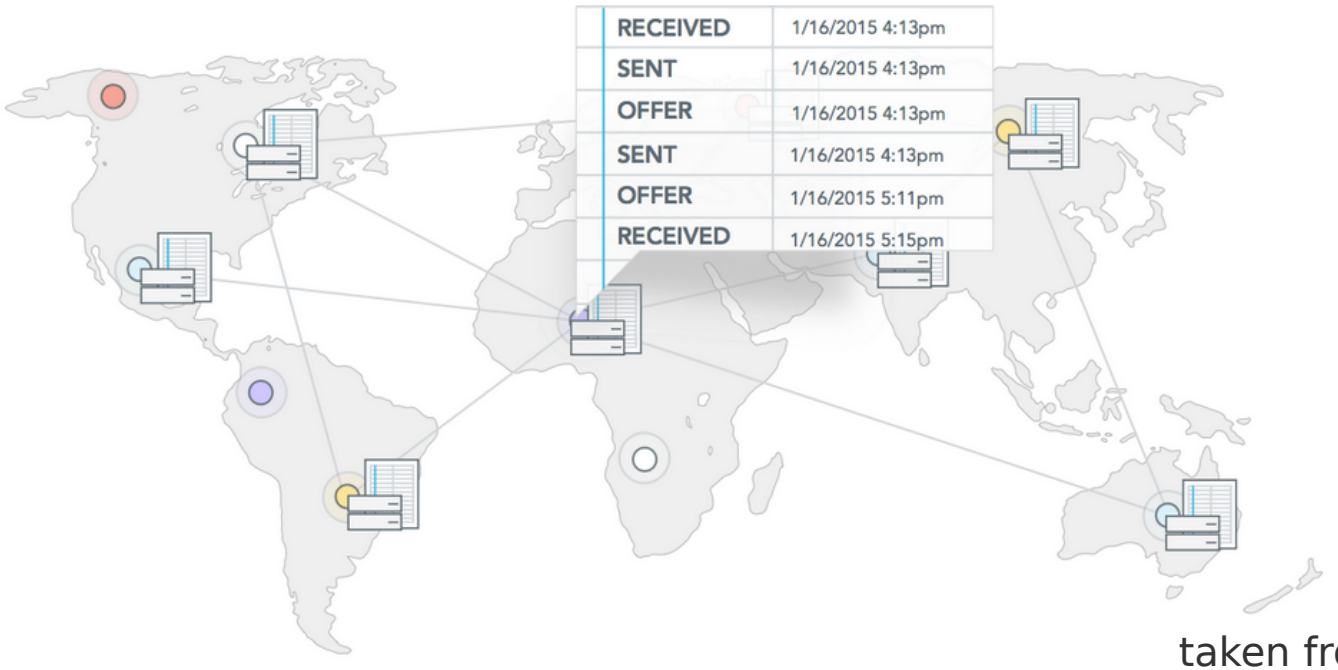
The Stellar Consensus Protocol

A Federated Model for Internet-level Consensus

ByzaRe, 2018-03-15, mail@maria-a-schett.net

Motivation: Stellar [1]

“Move Money Across Borders Quickly, Reliably, And For Fractions Of A Penny.”



taken from [1]

Stellar is a platform that connects banks, payments systems, and people.

What will I not talk about?

- Stellar.org vs. Stellar Network
  - nonprofit organization vs. technology, which is open source, distributed & community owned
- Lumens (XLM)
  - built-in currency for anti-spam & multi-currency bridge
- “integration” ~ connecting to Stellar network
  - software, tools & documentation
  - Apache License, version 2.0.
  - permits commercial use, modification, distribution

What will I not talk about?

- compliance and regulation
- KYC/AML identity verification

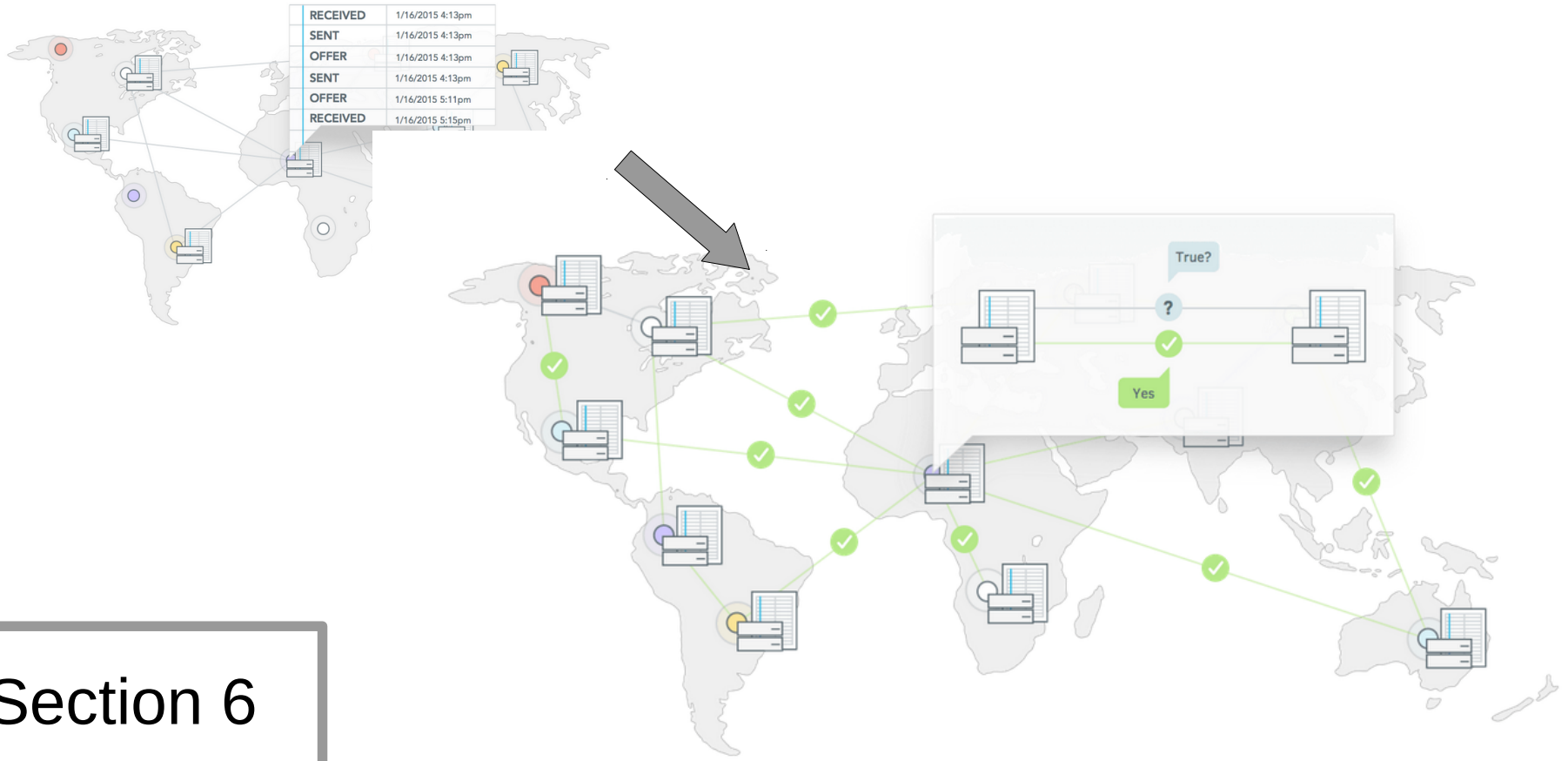
- for all this: see, e.g., [1]

stellar

What will I talk about?

taken from [2]

What will I talk about?

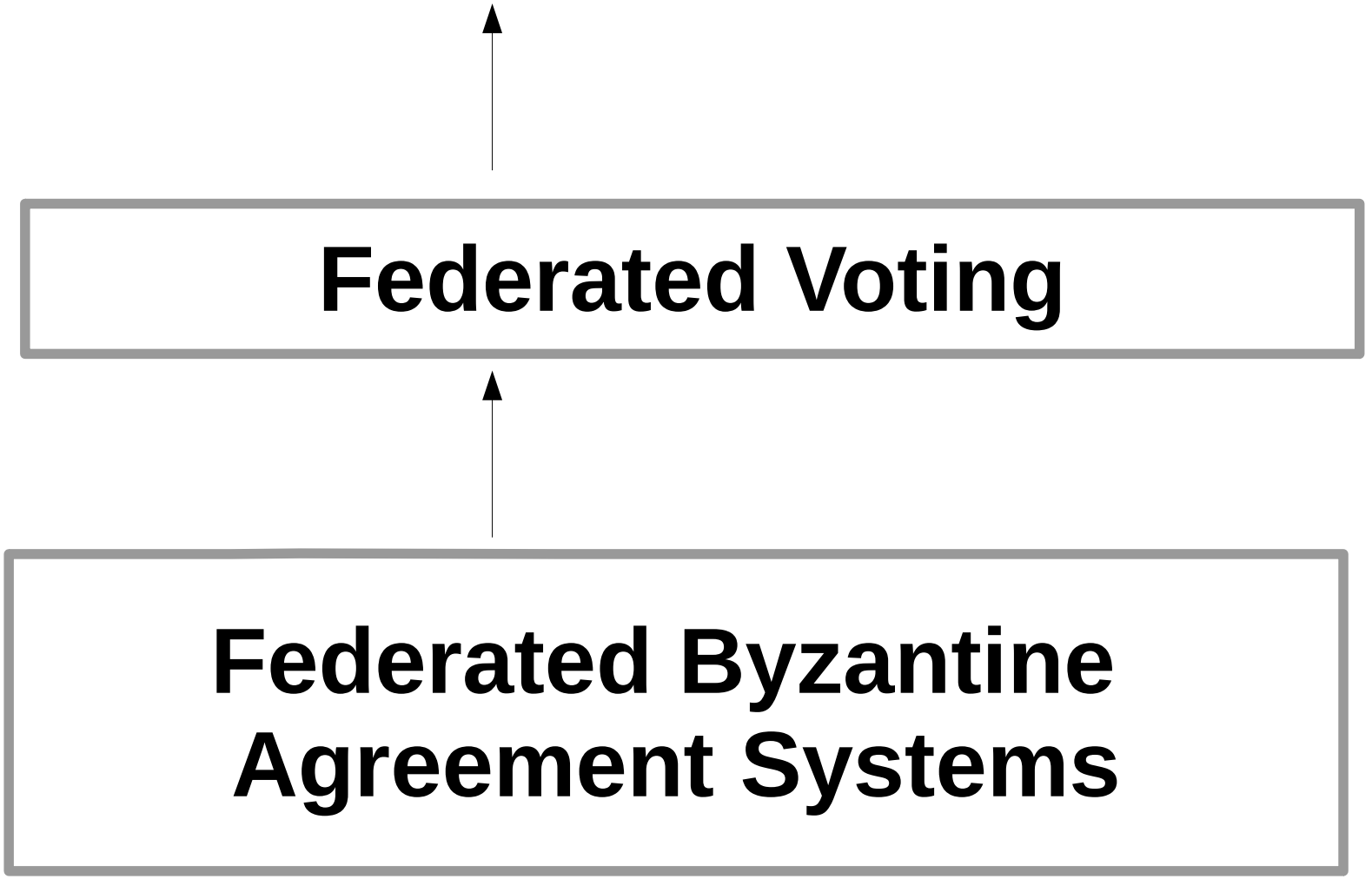


Section 6

taken from [1]

Prof. David Mazières's white paper,
The Stellar Consensus Protocol (SCP) [3]

Stellar Consensus Protocol



FBAS

- open membership

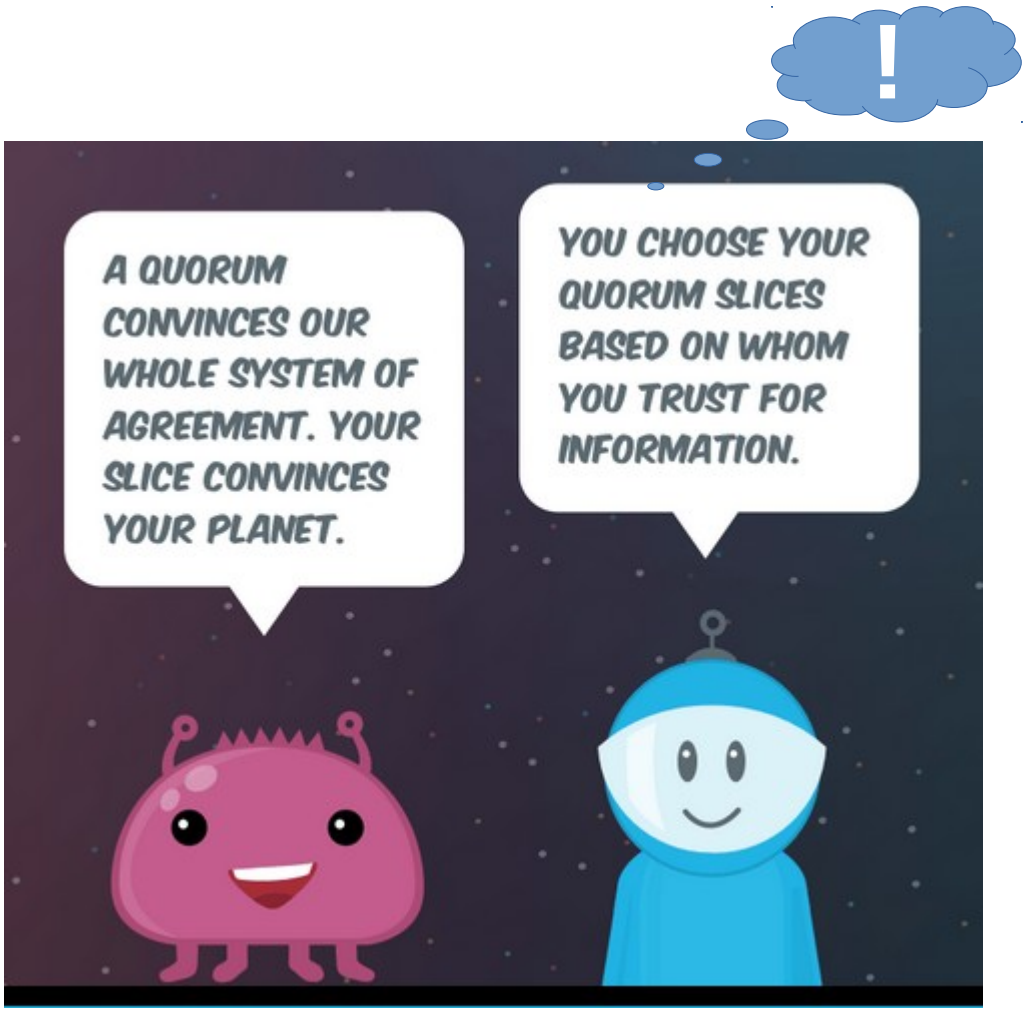
  → majority-based quorums do not work

- Federated Byzantine Agreement Systems:

  <V, Q : V → 2^2^V>

quorum slices

Quorum & Quorum Slice

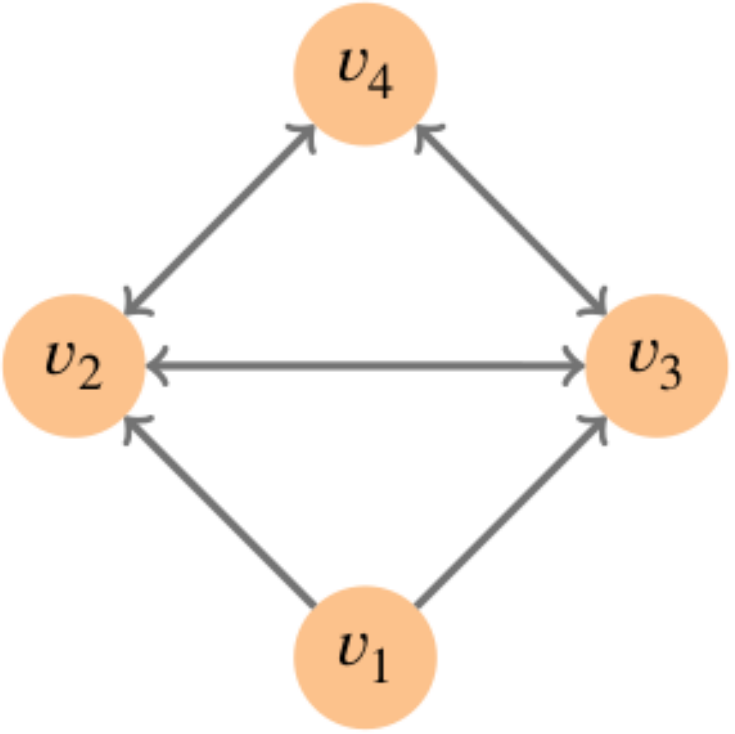


taken from [2]

set of nodes U is a **quorum** in FBAS <V, Q> if U contains a slice for each member, i.e.,

$\forall v \in U , \exists q \in Q(v)$ such that $q \subseteq U$

Quorum



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

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

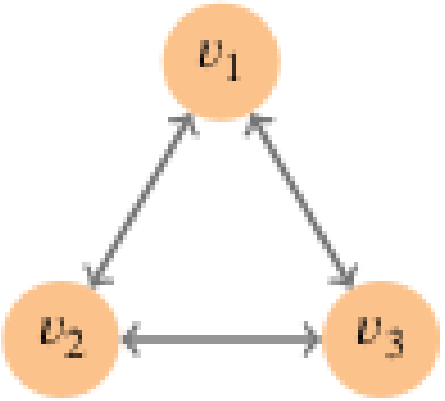
Fig. 2. v_1's quorum slice is not a quorum without v_4.

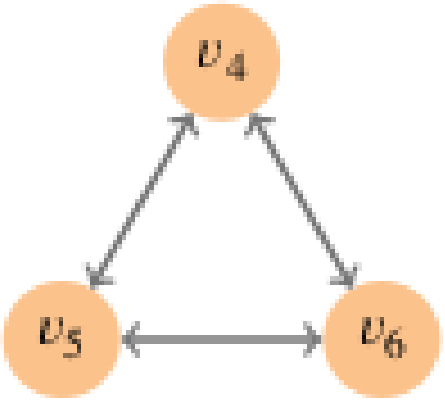
- only quorum including v_1 is V

Quorum Intersection

An FBAS enjoys quorum intersection iff any two of its quorums share a node.

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





$\mathbf{Q}(v_4) = \mathbf{Q}(v_5) = \mathbf{Q}(v_6) = \{\{v_4, v_5, v_6\}\}$

Fig. 6. FBAS lacking quorum intersection

Quorum Intersection

An FBAS enjoys quorum intersection iff any two of its quorums share a node.

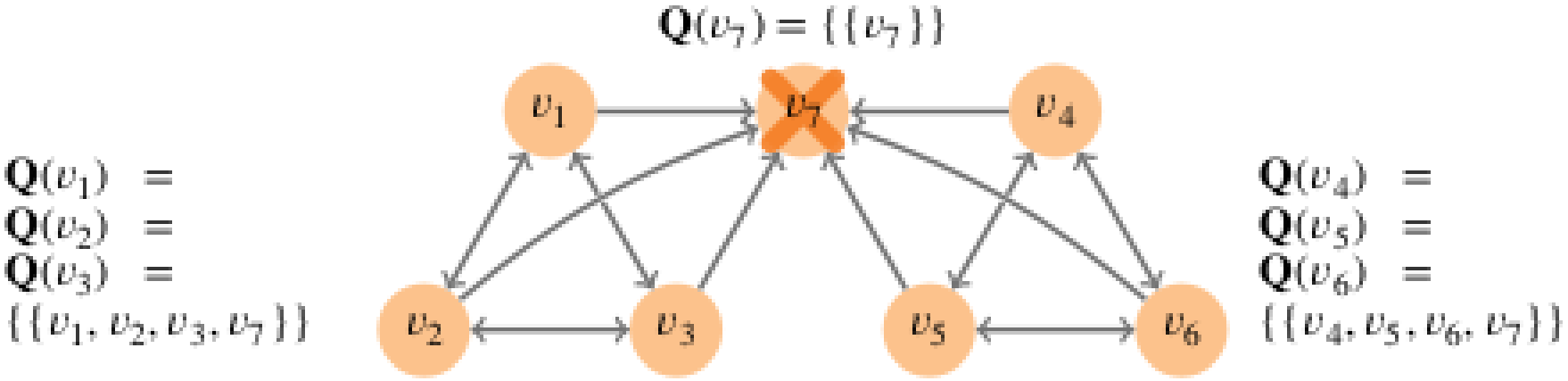


Fig. 7. Ill-behaved node v_7 can undermine quorum intersection.

DSets

- capture fault tolerance of nodes' slice selections through **dispensible set (DSet)**
- ***idea:*** safety and liveness for nodes outside a DSet can be guaranteed despite behaviour of nodes in DSet

- → want quorum intersection despite B and quorum availability despite B.

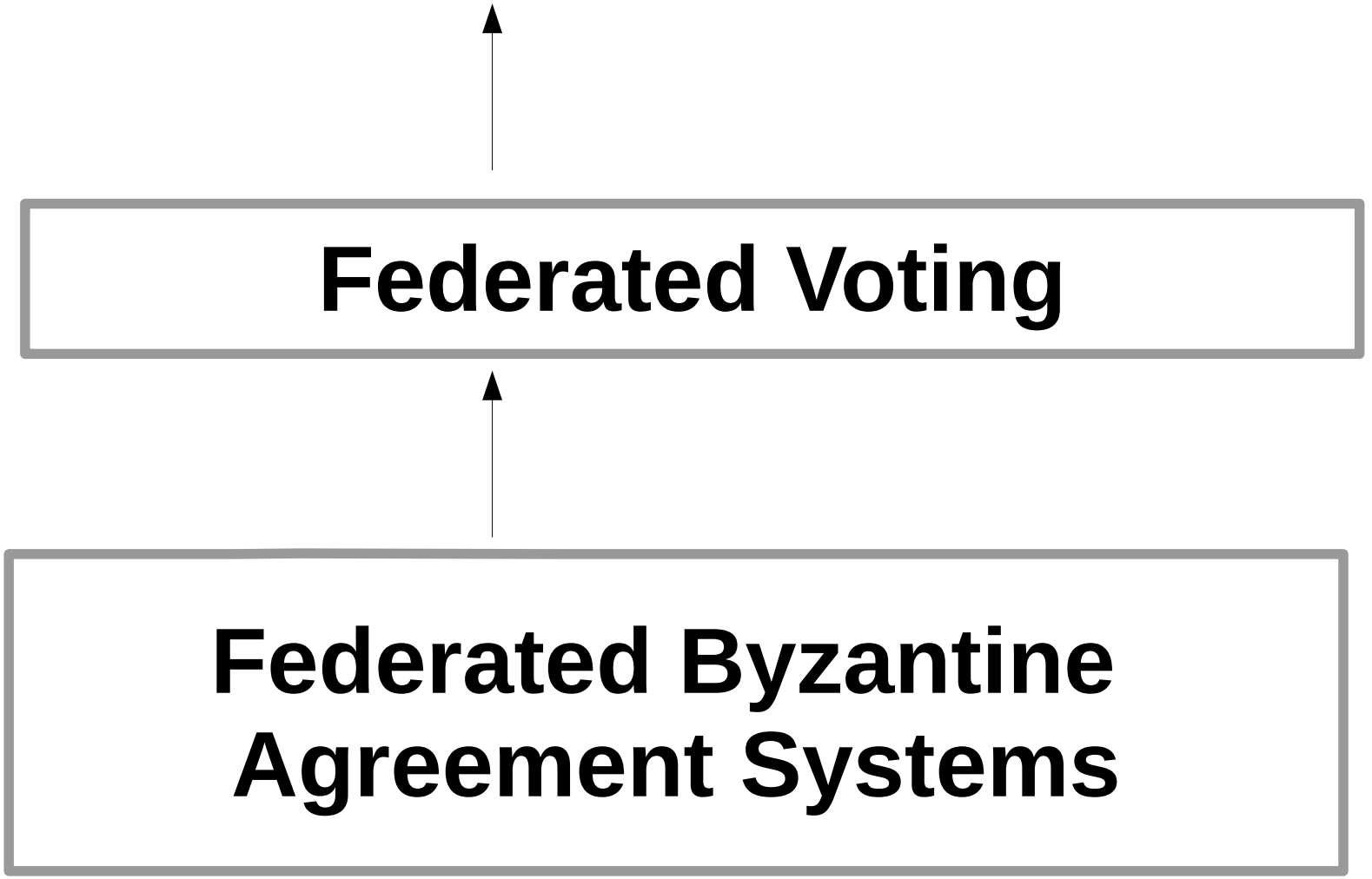
Theorem 3

In an FBAS with quorum intersection, the set of befouled nodes is a DSet.

Recap

- FBAS <V, Q>
- **quorum** in FBAS <V, Q> if U contains a slice for each member
- quorum intersection
- DSets

Stellar Consensus Protocol



Federated Voting

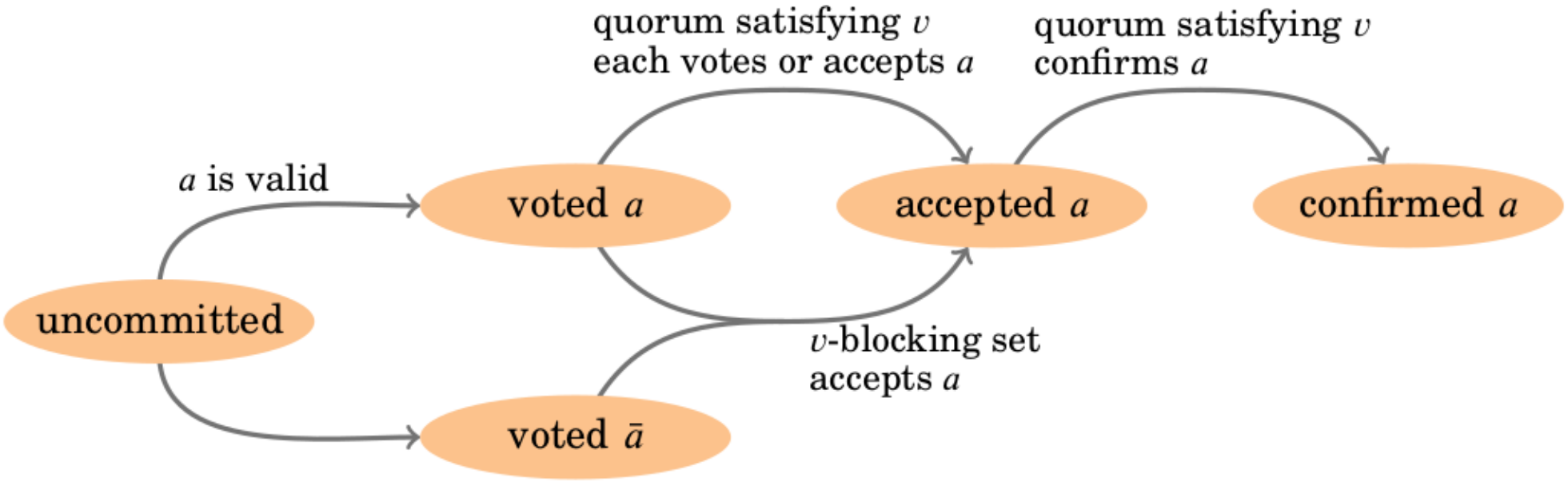


Fig. 11. Possible states of an accepted statement a at a single node v

- Example: lunch consensus [4]:

Hamburger or Falafel?

Federated Voting

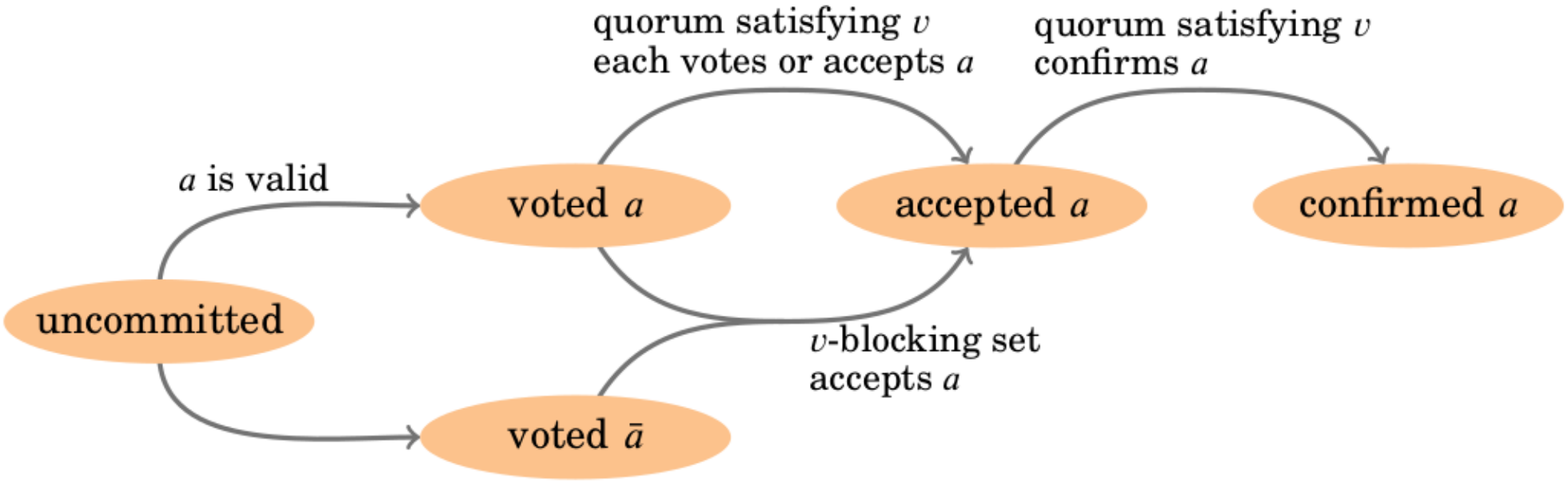


Fig. 11. Possible states of an accepted statement a at a single node v

taken from [3]

- **vote** Falafel ≈ remain open to possibility of accepting Falafel & promise to not vote for any option contradicting Falafel

Federated Voting

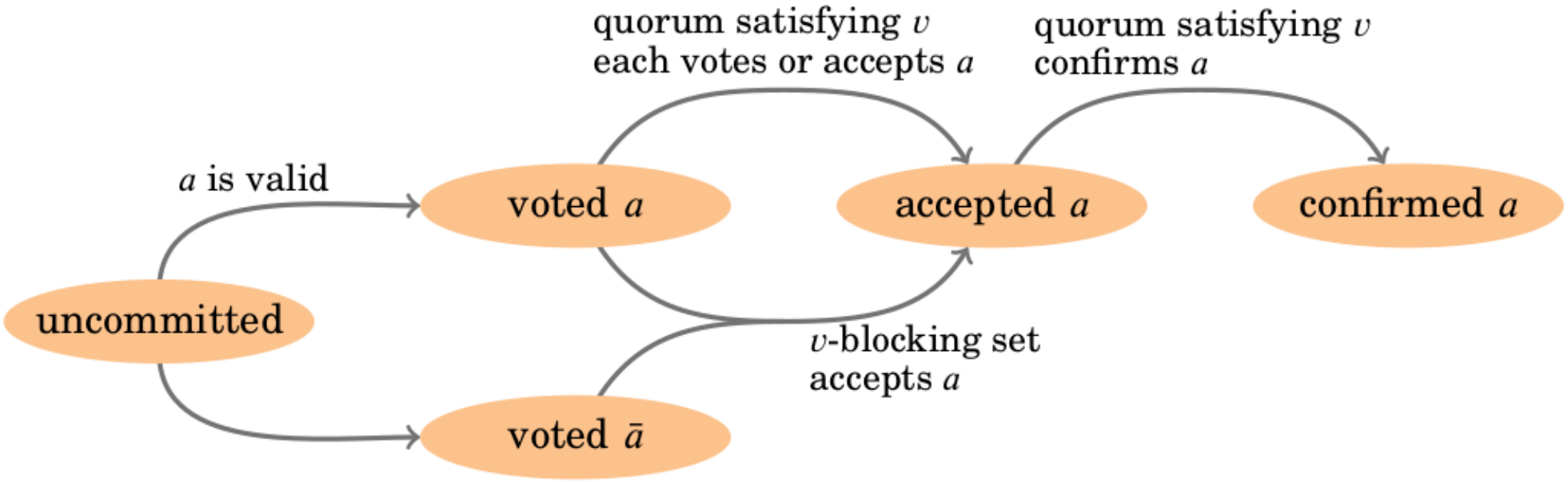


Fig. 11. Possible states of an accepted statement a at a single node v

- quorum U **ratifies** Falafel iff every member of U votes for Falafel

Ratify?

Theorem 4. Statements a and $\overline{a}$ cannot both be ratified in an FBAS that enjoys quorum intersection and contains no ill-behaved nodes.

- Proof. Contradiction to quorum intersection & assumption to no ill-behaved nodes.

Idea: collect all ill-behaved nodes in (a DSet) B

- **Theorem 6.** Two intact nodes in an FBA$\underline{S}$ with quorum intersection cannot ratify a and $\overline{a}$.

Federated Voting

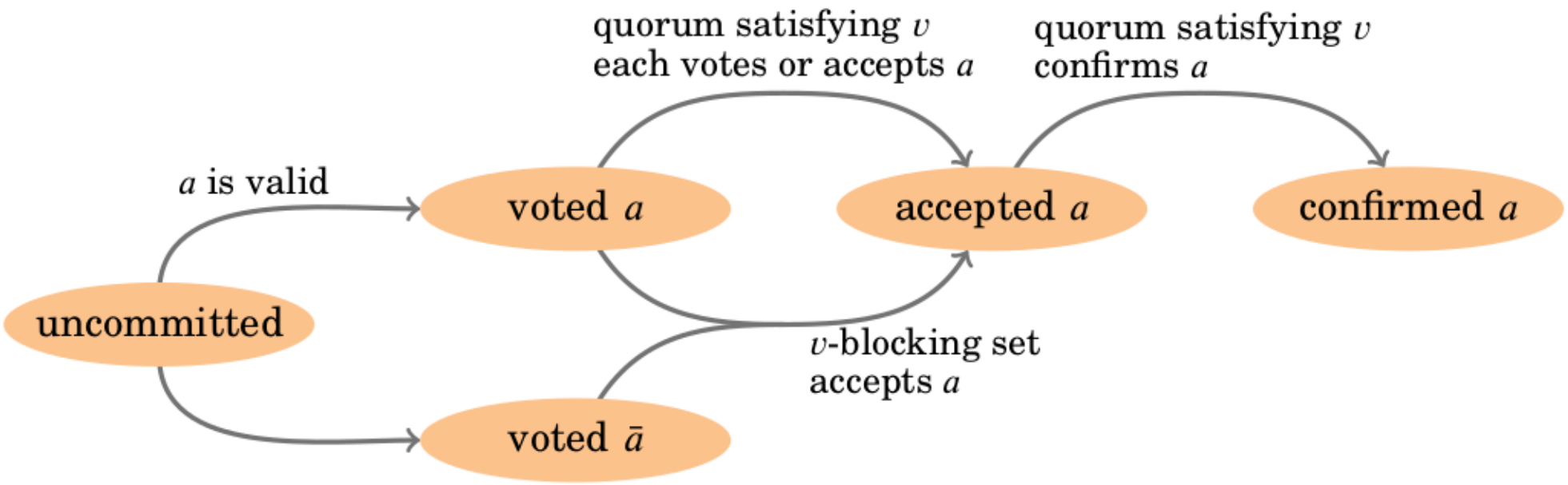


Fig. 11. Possible states of an accepted statement a at a single node v

taken from [3]

- **vote** Falafel → **accept** Falafel
- **vote** Hamburger → ?

But...

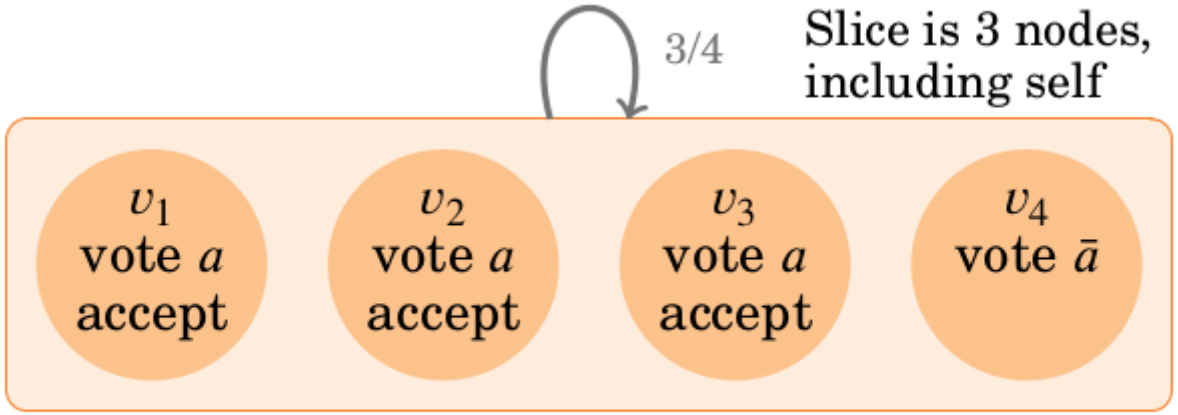


Fig. 9. v_4 voted for $\bar{a}$, which contradicts ratified statement a.

- **vote** “Hamburger”, but **accept** “Falafel”

v-blocking Sets

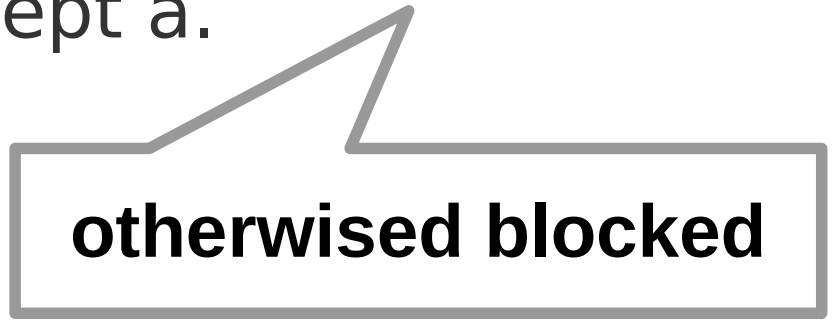
- could block v from making progress

A set $B \subseteq V$ is **v-blocking** iff it overlaps every one of v's slices—i.e., $\forall q \in Q(v), q \cap B \neq \emptyset$

Accept

v **accepts** a iff it has never accepted a stmt contradicting a and

- (1) There exists a quorum U such that v ∈ U and each member of U either voted for a or claims to accept a, or
- (2) Each member of a v-blocking set claims to accept a.



All intact nodes can accept?

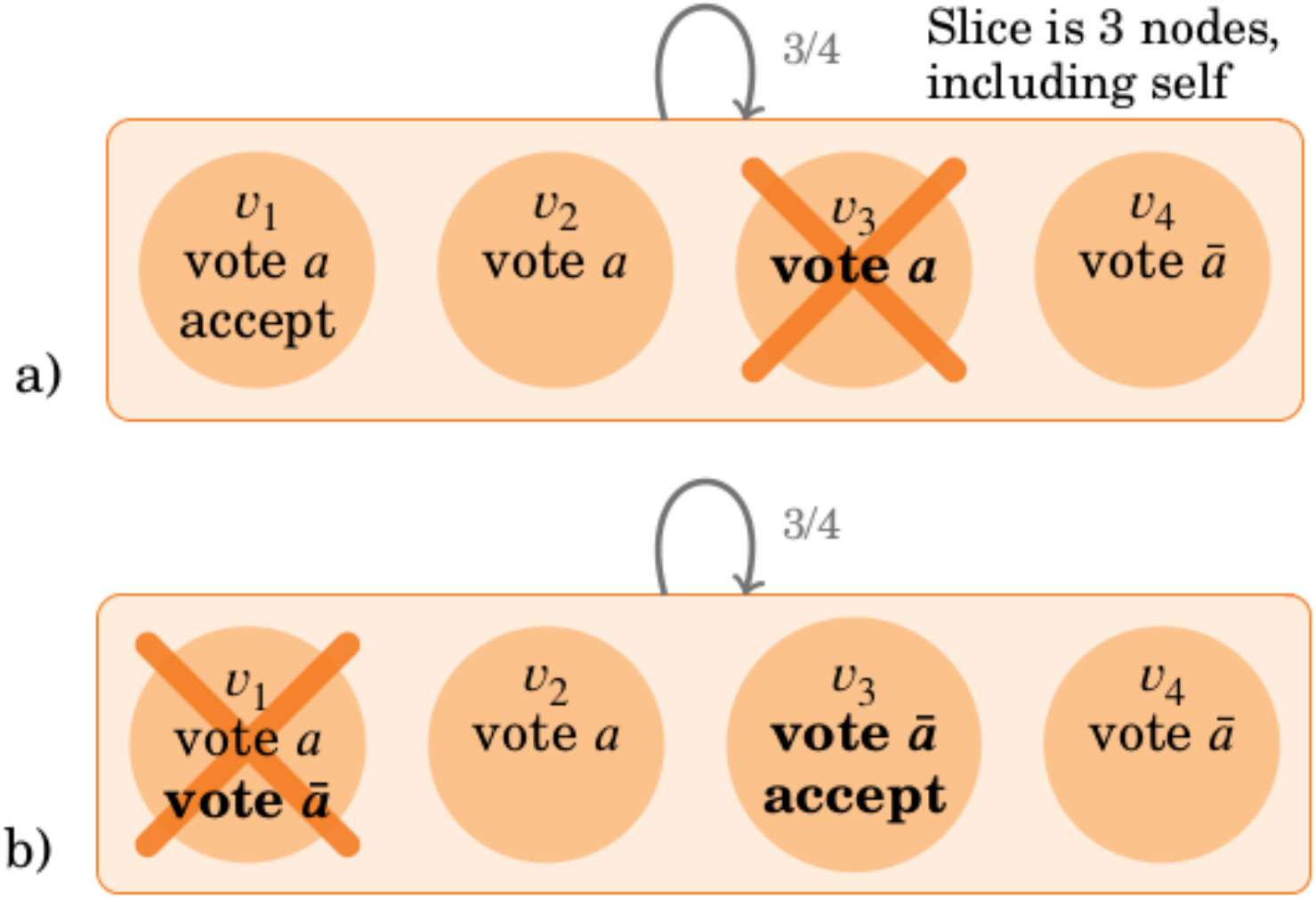


Fig. 10. Scenarios indistinguishable to v_2 when v_2 does not see bold messages

Federated Voting

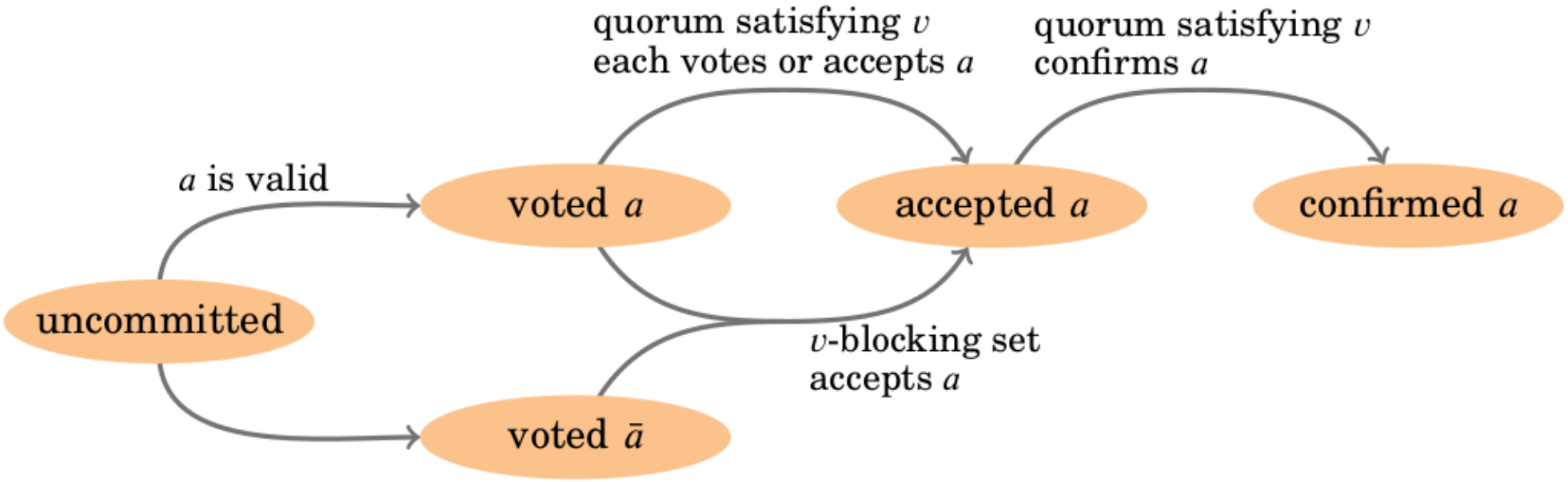


Fig. 11. Possible states of an accepted statement a at a single node v

taken from [3]

- **confirm**

Confirm

- **Theorem 11.** If an intact node in an FBAS with quorum intersection confirms a statement a, then, whatever subsequently transpires, once sufficient messages are delivered and processed, every intact node will accept and confirm a.

Recap

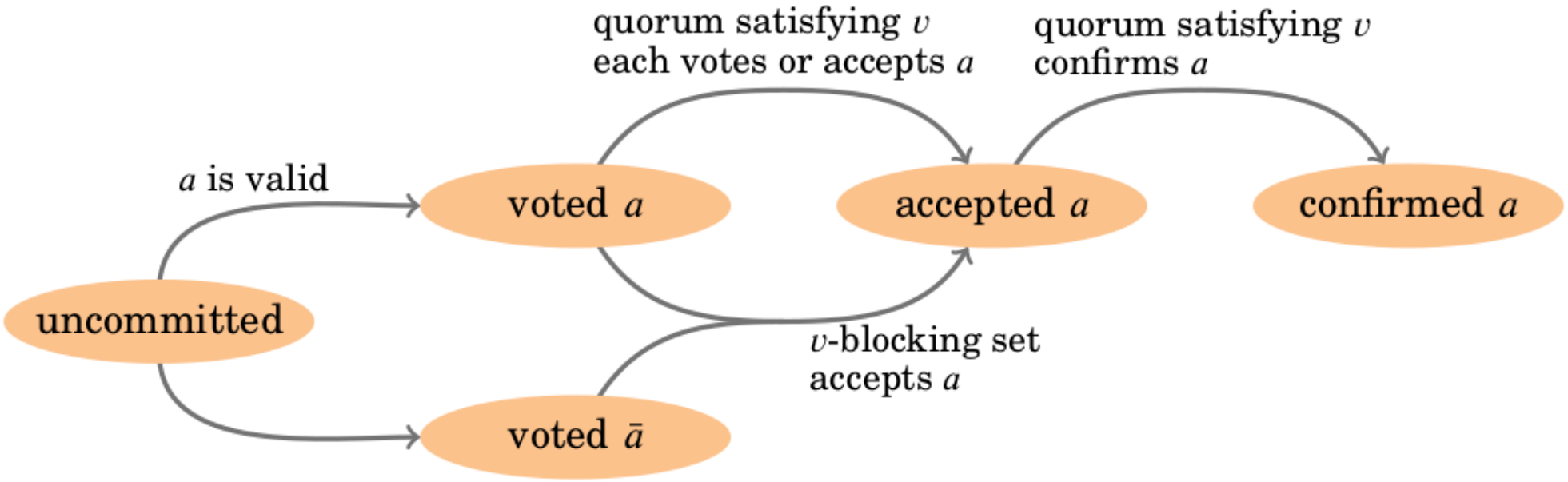
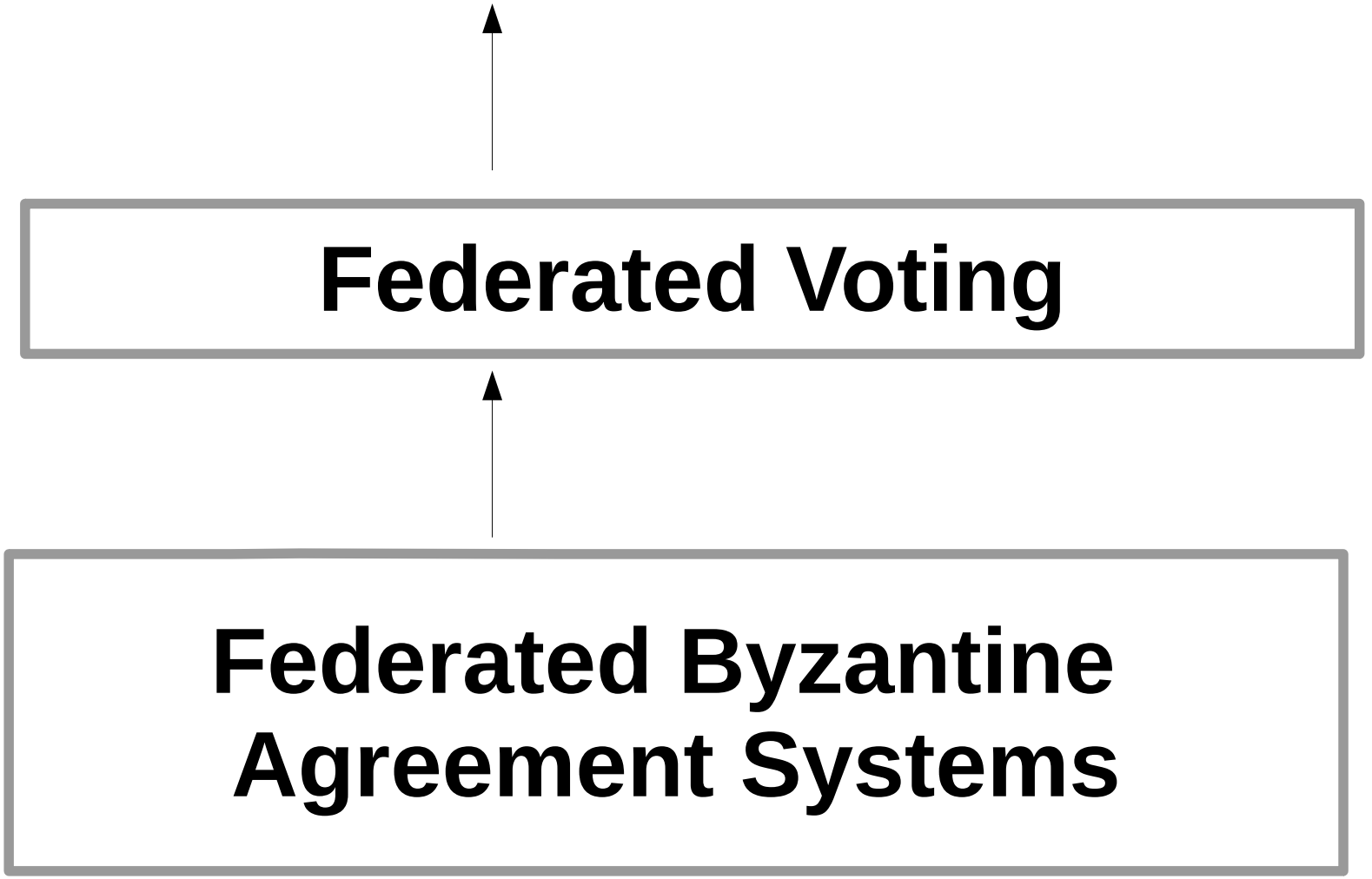


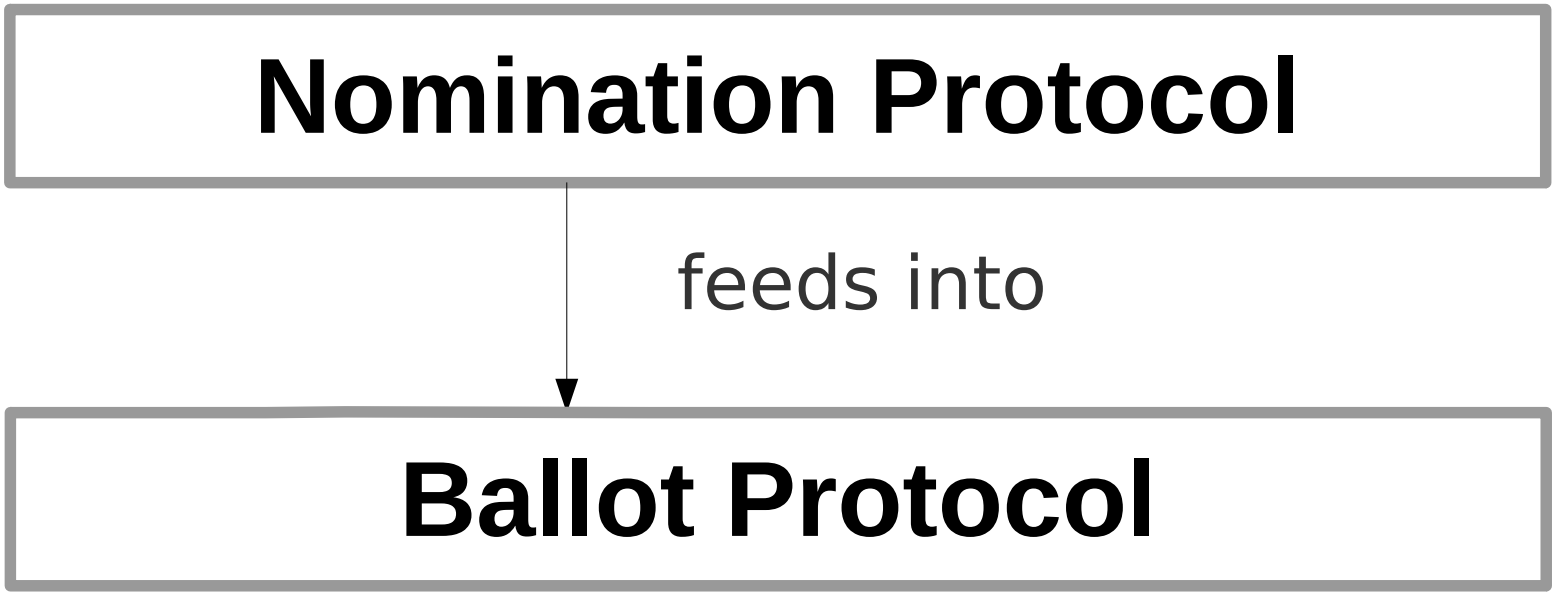
Fig. 11. Possible states of an accepted statement a at a single node v

Stellar Consensus Protocol



Stellar Consensus Protocol

- synchronous



Nomination Protocol

- get at least one candidate value for each slot

  → by federated voting on stmt “nominate x”

  → deterministic computation of

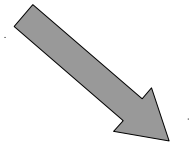
  composite values

- X … set of values v has voted to nominate
- Y … set of values v has accepted as nominated
- Z … set of values v considers candidate values

Nomination Protocol

Theorem 12. Eventually, all intact nodes will have the same composite value.

- When converged?
- Can Byzantine nodes stop that?



“when nodes guess convergence, start **ballot protocol**”

Ballot Protocol

- ballot = <n, x>

  total order on ballots

  → federated voting on stmt “commit b”

  or “$\overline{\text{commit b}}$” ~ “abort b”

- ballot b is **prepared** iff every statement in the following set is true: { abort b_{old} | $b_{old} \lesssim b$ }

Ballot Protocol

- → federated voting on stmt “commit b”
  or “$\overline{\text{commit b}}$” ~ “abort b”

- 3 Phases: **prepare**, **confirm**, **externalize**

Receive msg m → v updates

(1) If $\varphi = \text{PREPARE}$ and m lets v accept new ballots as prepared, update p and p'. Afterwards, if either $p \not\gtrsim h$ or $p' \not\gtrsim h$, then set $c \leftarrow \mathbf{0}$.

(2) If $\varphi = \text{PREPARE}$ and m lets v confirm new higher ballots prepared, then raise h to the highest such ballot and set $z \leftarrow h.x$.

(3) If $\varphi = \text{PREPARE}$, $c = \mathbf{0}$, $b \leq h$, and neither $p \gtrsim h$ nor $p' \gtrsim h$, then set c to the lowest ballot satisfying $b \leq c \lesssim h$.

(4) If $\varphi = \text{PREPARE}$ and v accepts *commit* for one or more ballots, set c to the lowest such ballot, then set h to the highest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid c \lesssim b' \lesssim h \}$, and set $\varphi \leftarrow \text{CONFIRM}$. Also set $z \leftarrow h.x$ after updating h, and unless $h \lesssim b$, set $b \leftarrow h$.

(5) If $\varphi = \text{CONFIRM}$ and the received message lets v accept new ballots prepared, raise p to the highest accepted prepared ballot such that $p \sim c$.

(6) If $\varphi = \text{CONFIRM}$ and v accepts more *commit* messages or raises b, then let h' be the highest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid b \lesssim b' \lesssim h' \}$ (if any). If there exists such an h' and $h' > h$, then set $h \leftarrow h'$, and, if necessary, raise c to the lowest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid c \lesssim b' \lesssim h \}$.

(7) If $\varphi = \text{CONFIRM}$ and v confirms *commit* c' for any c', set c and h to the lowest and highest such ballots, set $\varphi \leftarrow \text{EXTERNALIZE}$, externalize $c.x$, and terminate.

(8) If $\varphi \in \{\text{PREPARE}, \text{CONFIRM}\}$ and $b < h$, then set $b \leftarrow h$.

(9) If $\varphi \in \{\text{PREPARE}, \text{CONFIRM}\}$ and $\exists S \subseteq M_v$ such that the set of senders $\{ v_{m'} \mid m' \in S \}$ is v-blocking and $\forall m' \in S$, $b_{m'}.n > b_v.n$, then set $b \leftarrow \langle n, z \rangle$, where n is the lowest counter for which no such S exists. Repeat the previous steps after updating b.

Receive msg m → v updates

(1) If φ = PREPARE and m lets v accept new ballots as prepared, update p and p'. Afterwards, if either $p \not\gtrsim h$ or $p' \not\gtrsim h$, then set $c \leftarrow \mathbf{0}$.

(2) If φ = PREPARE and m lets v confirm new higher ballots prepared, then raise h to the highest such ballot and set $z \leftarrow h.x$.

(3) If φ = PREPARE, $c = \mathbf{0}$, $b \lesssim h$, and neither $p \gtrsim h$ nor $p' \gtrsim h$, then set c to the lowest ballot satisfying $b \lesssim c \lesssim h$.

(4) If φ = PREPARE and v accepts *commit* for one or more ballots, set c to the lowest such ballot, then set h to the highest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid c \lesssim b' \lesssim h \}$, and set $\varphi \leftarrow$ CONFIRM. Also set $z \leftarrow h.x$ after updating h, and unless $h \lesssim b$, set $b \leftarrow h$.

(5) If φ = CONFIRM and the received message lets v accept new ballots prepared, raise p to the highest accepted prepared ballot such that $p \sim c$.

(6) If φ = CONFIRM and v accepts more *commit* messages or raises b, then let h' be the highest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid b \lesssim b' \lesssim h' \}$ (if any). If there exists such an h' and $h' > h$, then set $h \leftarrow h'$, and, if necessary, raise c to the lowest ballot such that v accepts all $\{ \mathit{commit}\ b' \mid c \lesssim b' \lesssim h \}$.

(7) If φ = CONFIRM and v confirms *commit* c' for any c', set c and h to the lowest and highest such… EXTERNALIZE, externalize $c.x$, and terminate.

(8) I…

(9) I…

only if nomination protocol converged, else time out and try again with higher ballot

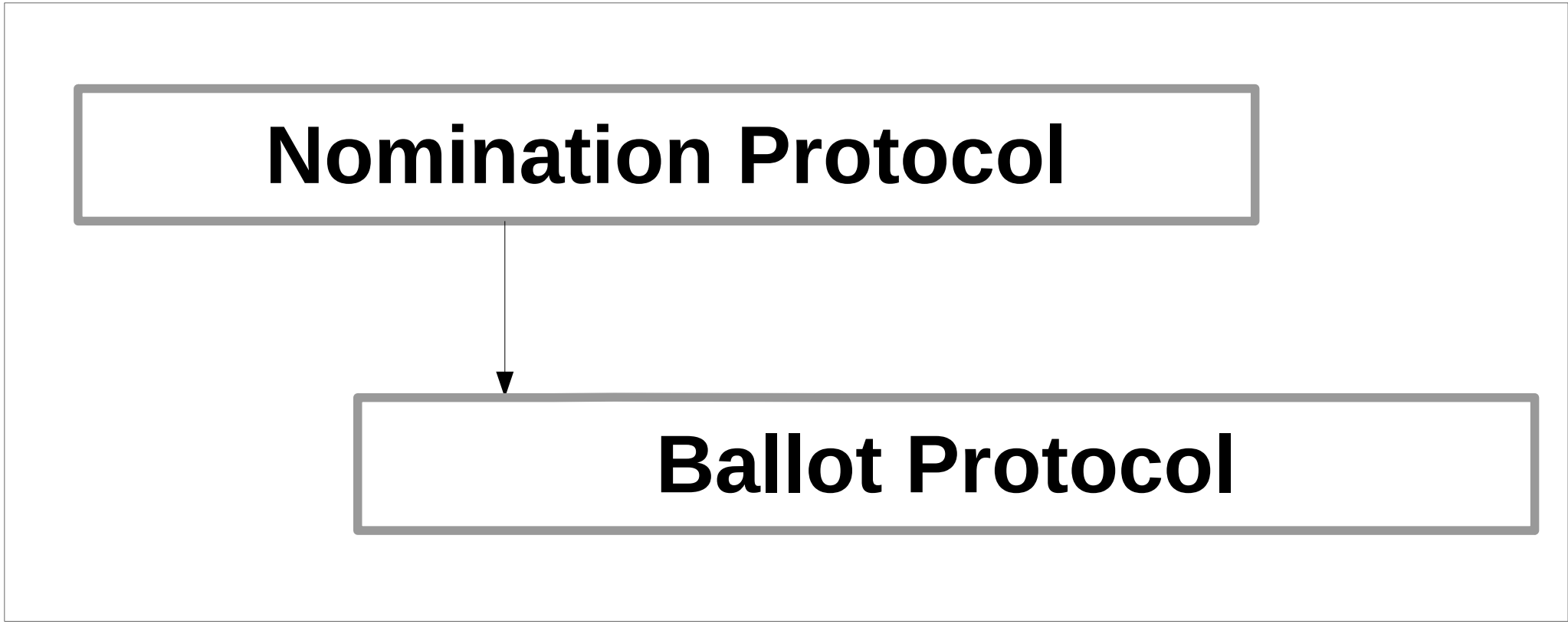
Correctness

Theorem 15. Given long enough timeouts, if an intact node has reached the CONFIRM phase with b.x = x, then eventually all intact nodes will terminate.

Theorem 16. Regardless of past ill-behavior, given long enough timeouts and periods in which ill-behaved nodes do not send new messages, intact nodes running SCP will terminate.

Recap

Stellar Consensus Protocol



Next Steps

- fully understand “big picture”
- (and some individual components)

  → formalize in Coq

Re-Sources

[1] https://www.stellar.org/how-it-works/stellar-basics/#how-it-works

[2] https://www.stellar.org/stories/adventures-in-galactic-consensus-chapter-1/

[3] Mazières, David. "The stellar consensus protocol: A federated model for internet-level consensus." Stellar Development Foundation (2015).

[4] https://medium.com/a-stellar-journey/on-worldwide-consensus-359e9eb3e949