

Good morning ↑
c

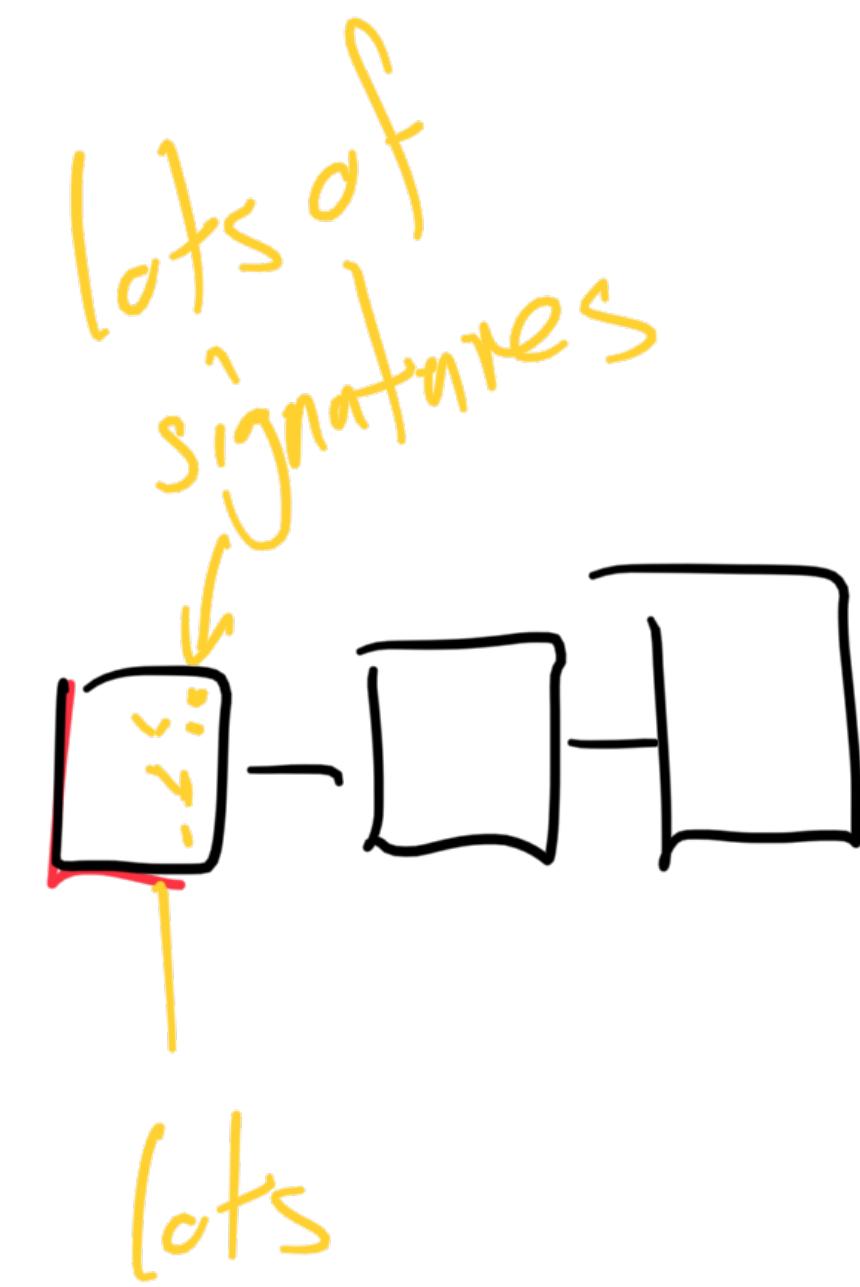
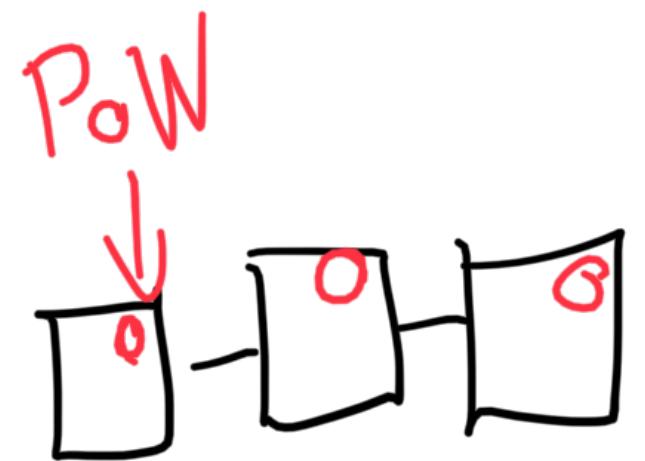
A BFT protocol

Simplified HotStuff

Leander Jehl

PoW or Certificates

Idea



BFT protocol

Certificate vs. PoW

PoW: Requiring that blocks contains a proof of work gives the following:

- **Rate limit:** Limit at which rate new blocks are created.
- **Fork probability:** Reduce probability for forks
- **Prevent system split:** Small subsystem will not be able to create blocks at correct rate.

BFT protocol

Certificate vs. PoW

Certificate: If blocks require a certificate, we get similar properties.

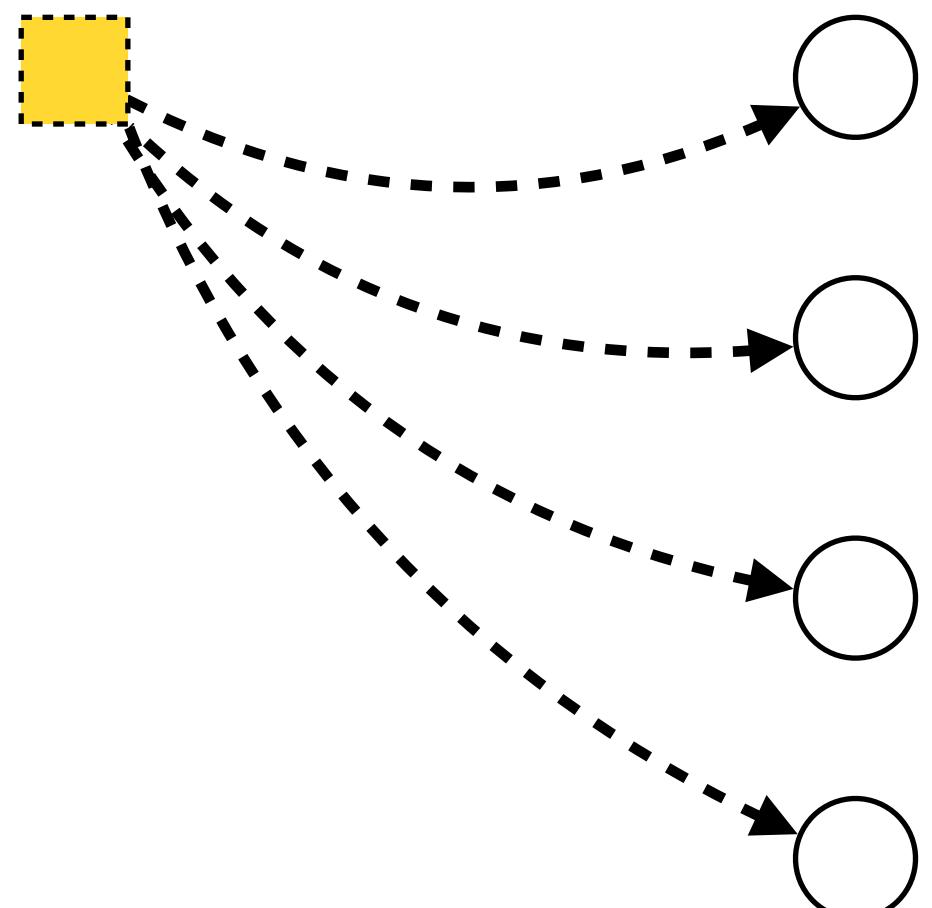
- **Rate limit:**
- **Fork probability:**
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

new block

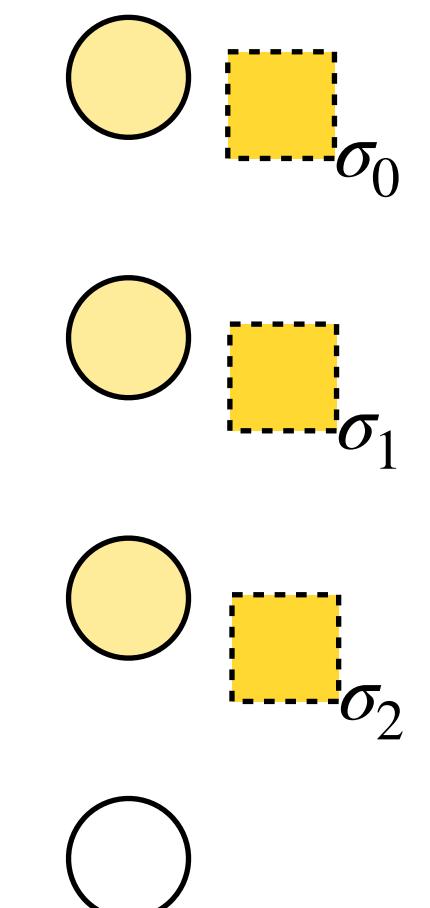


nodes

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

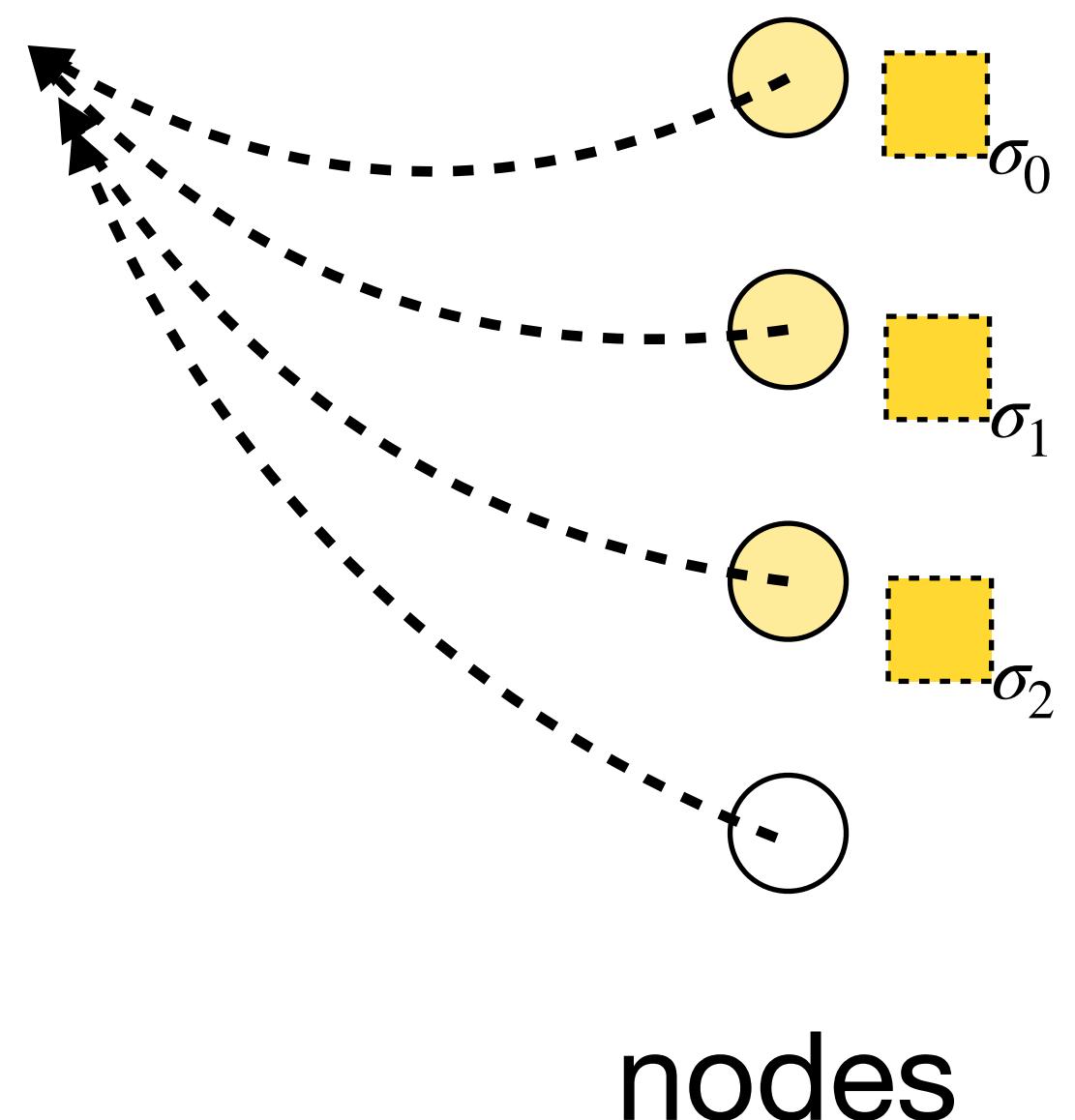


nodes

BFT protocol

Certificate vs. PoW

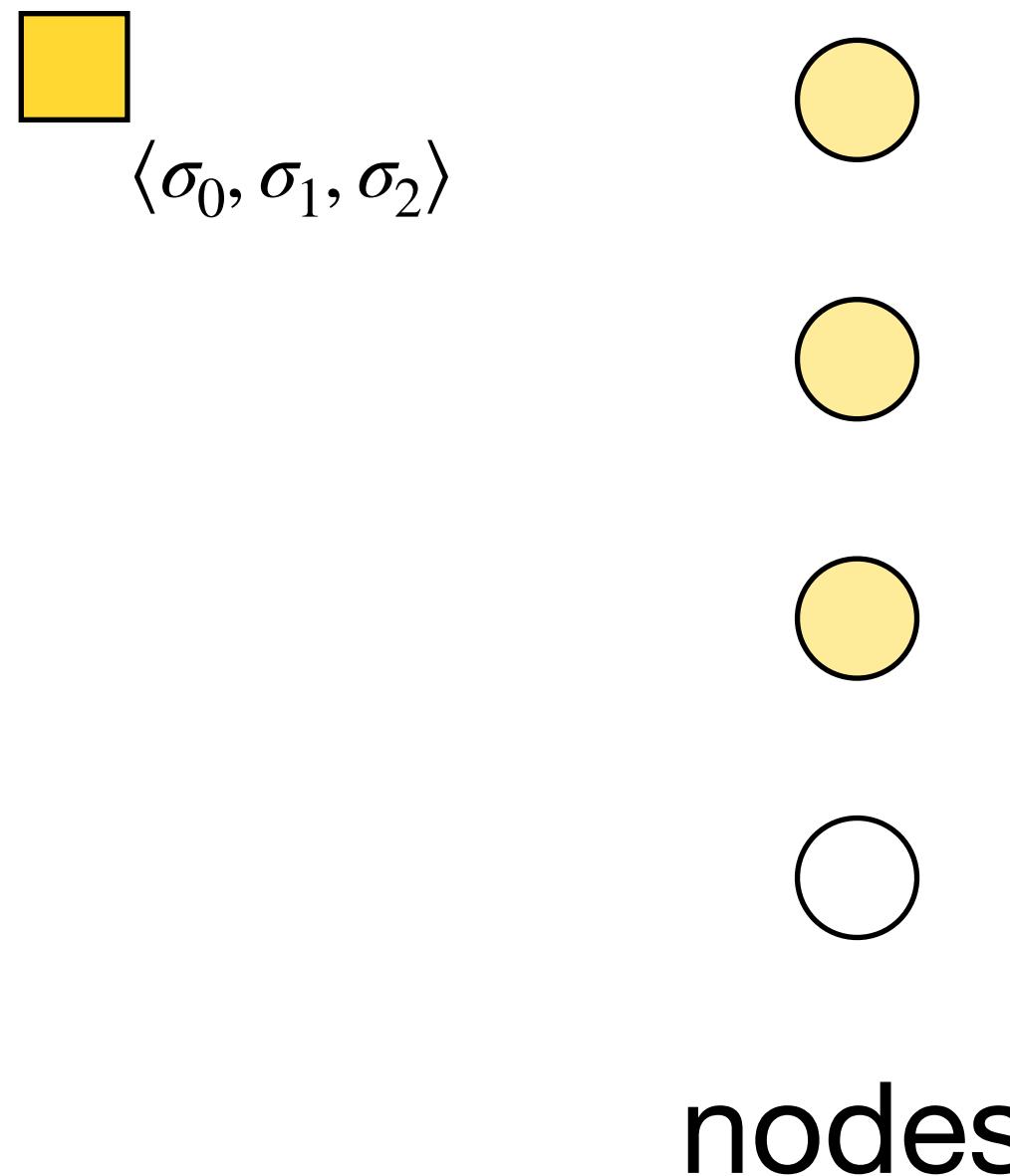
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

Certificate: If blocks require a certificate, we get similar properties.

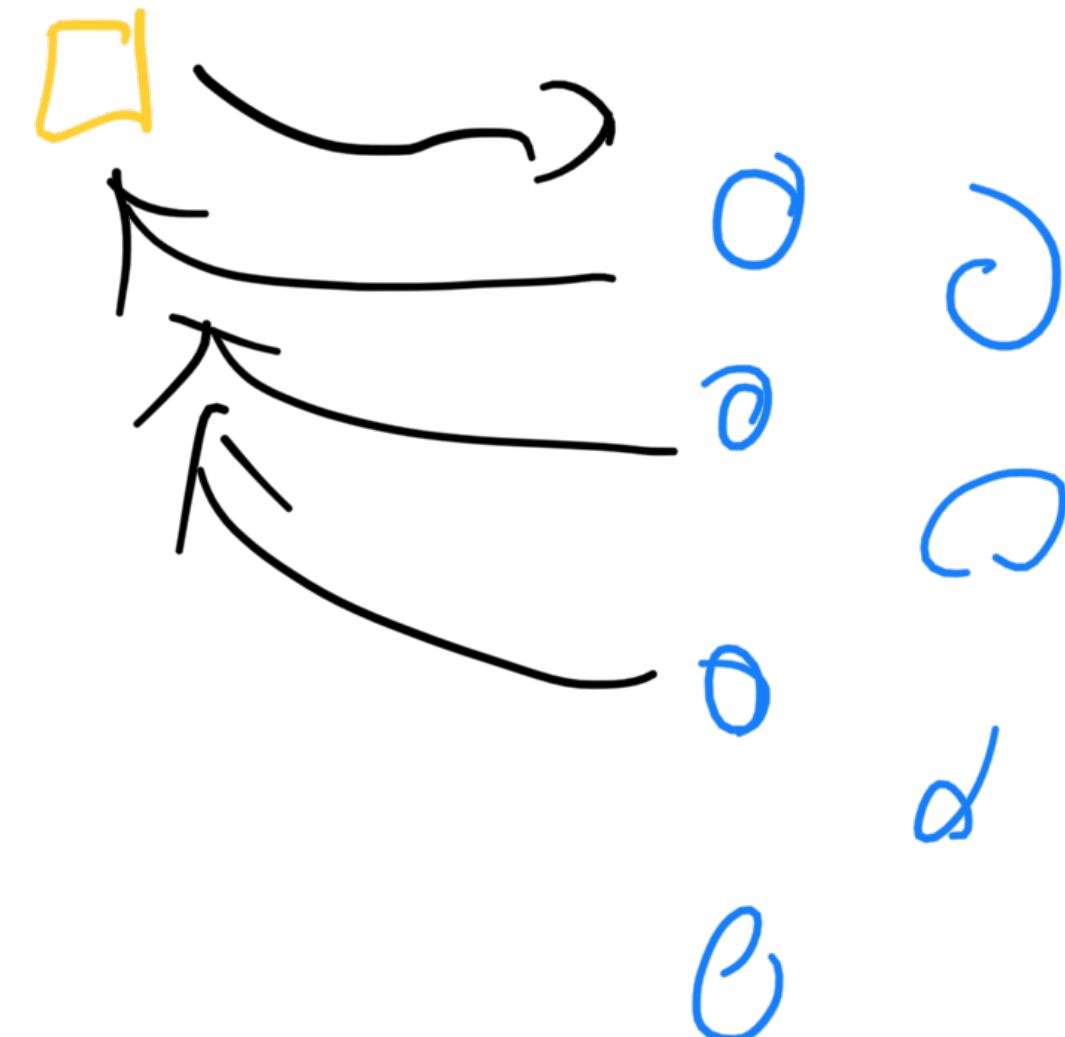
- **Rate limit:**
- **Fork probability:**
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

- **Rate limit:**



Need to wait for signatures.

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

Certificate: If blocks require a certificate, we get similar properties.

- **Rate limit:**
Blocks need to be verified and signed by most of the nodes.
Cannot create blocks faster than they are verified and signed.
- **Fork probability:**
- **Prevent system split:**

BFT protocol

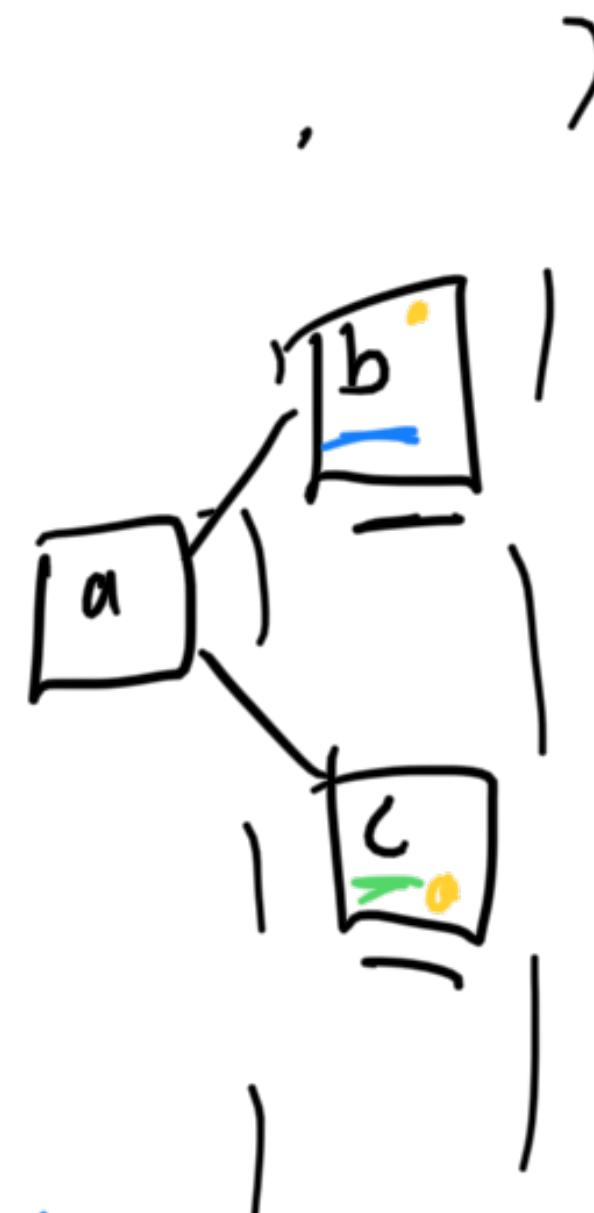
Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

- **Fork probability:**

How big is a certificate?

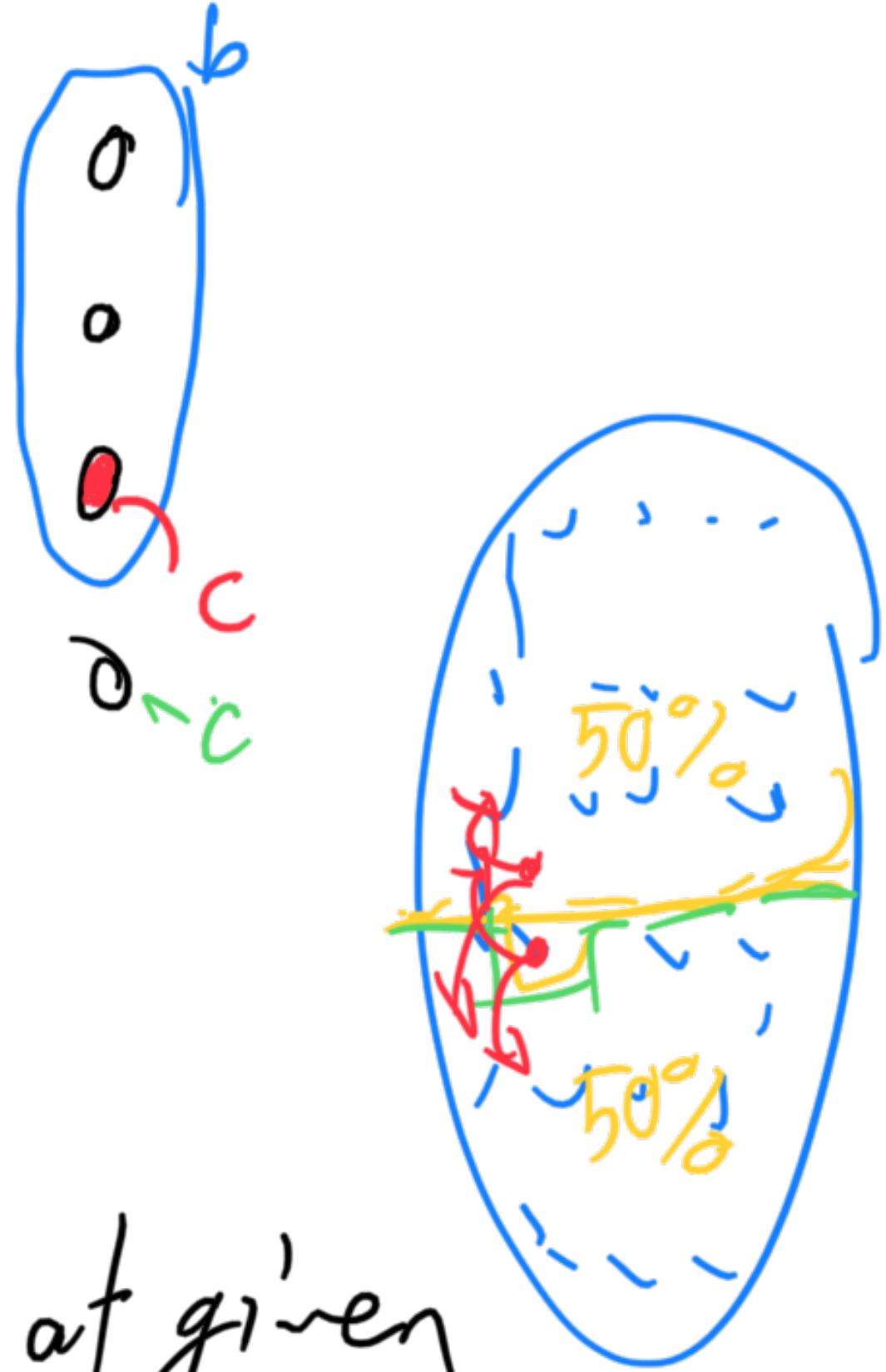
- small \rightarrow forks
- majority $> 50\%$ \sim unlikely
 $\bullet > \frac{2}{3}$ but possible
- all \sim forks unlikely, but cannot tolerate failures.



try to enforce,
• only sign once at given height.

67 of 100

6 no forks if at most
33 do double sign.
 \hookrightarrow can tolerate 33 outages.



BFT protocol

Model

Model:

- We assume a permissioned system with $\underline{N} = \underline{3f + 1}$ nodes.
- Nodes have unique ids and unique, known cryptographic keys.
- At most \underline{f} of the nodes are byzantine faulty.

Certificate:

- A block has a certificate, if it contains signatures of $\underline{2f + 1}$ nodes.

$$N = 4 = 3 \cdot 1 + 1$$

$$f = 1$$

$$\text{cert} = 3$$

$$N = 100 = 3 \cdot 33 + 1$$

$$f = 33$$

$$\text{cert } 67$$

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

- **Fork probability:**

BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.

Certificate: If blocks require a certificate, we get similar properties.

- **Rate limit:**
- **Fork probability:**
If nodes do not sign multiple blocks, at most one block at a given height can get a certificate.
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

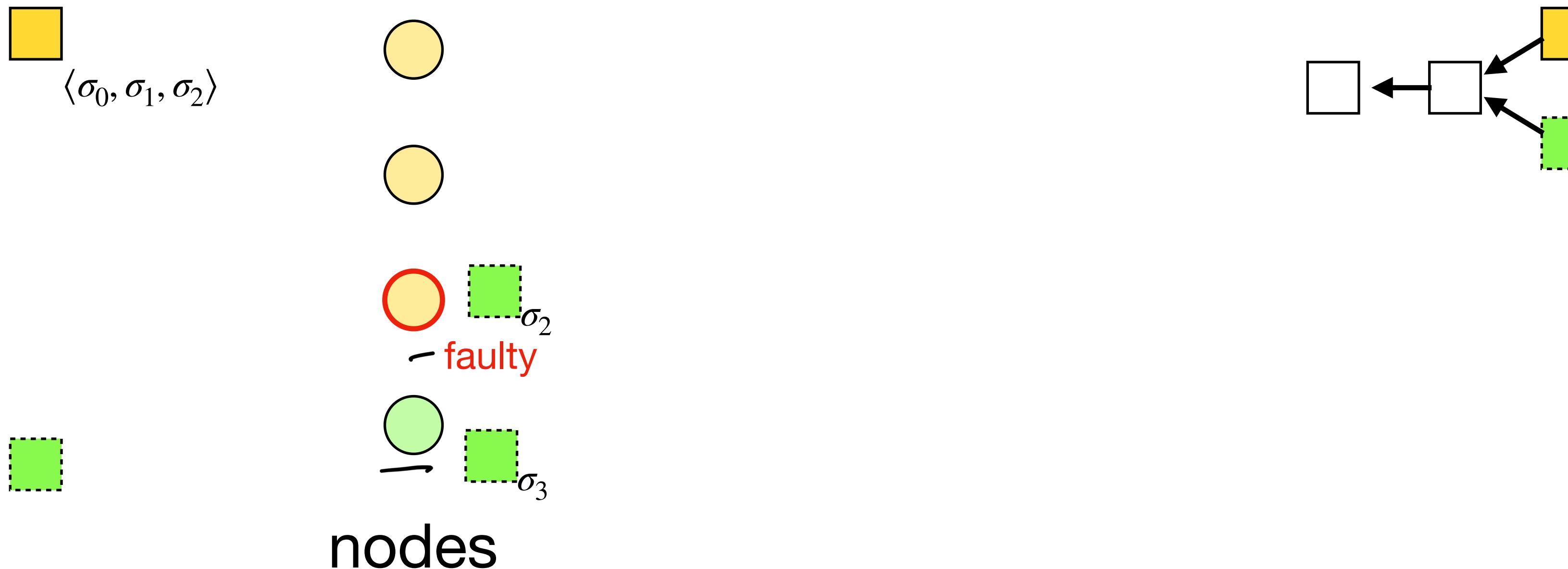
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

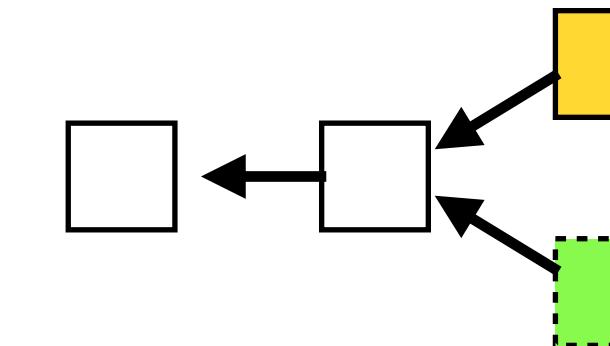
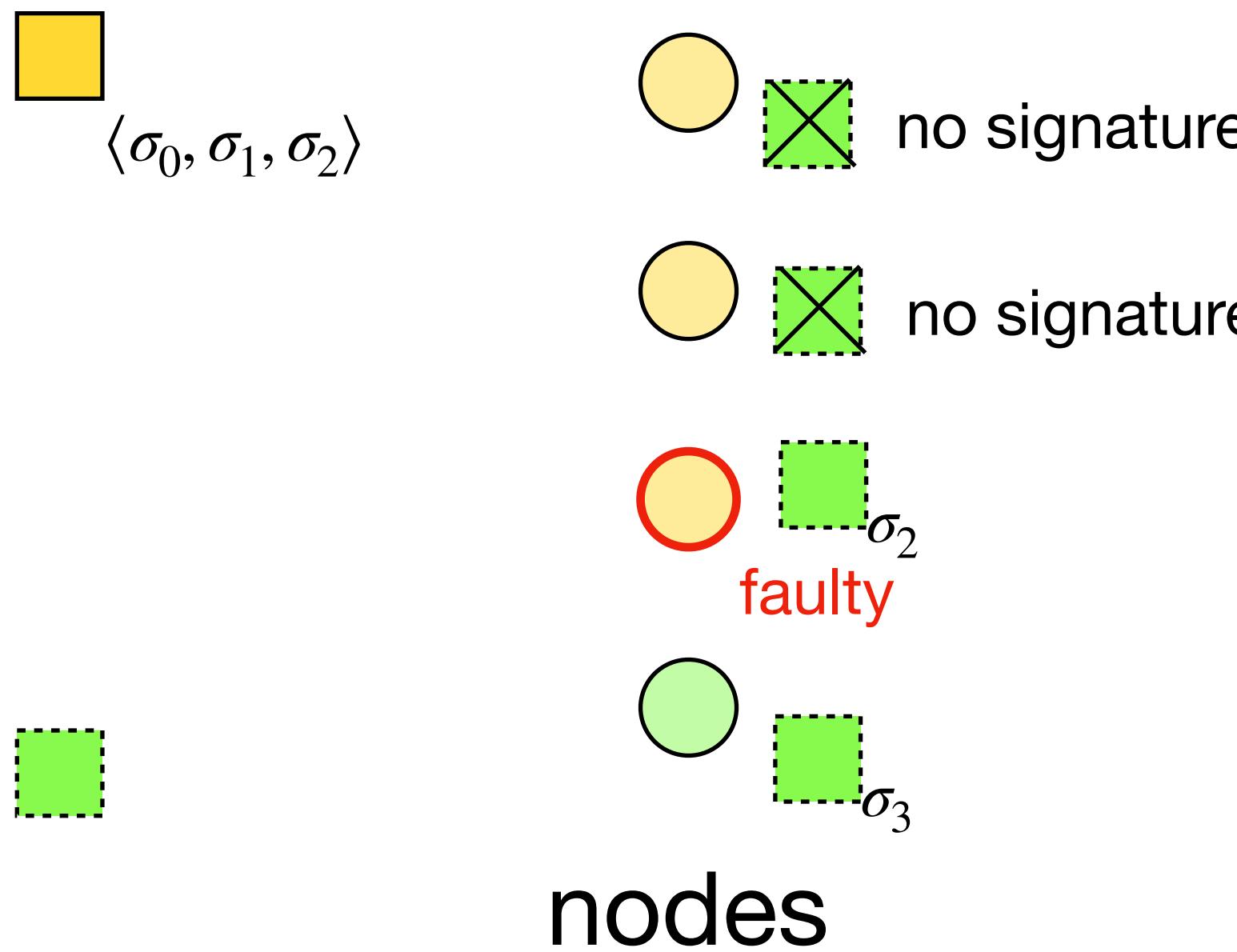
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

Idea: Send new block to nodes for validation and signature.

Correct nodes sign only one block at given depth.

Then collect certificate.

Certificate: If blocks require a certificate, we get similar properties.

- **Rate limit:**
 - **Fork probability:**
 - If nodes do not sign multiple blocks, at most one block at a given height can get a certificate.
- Obs: Faulty nodes may sign multiple blocks!**
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

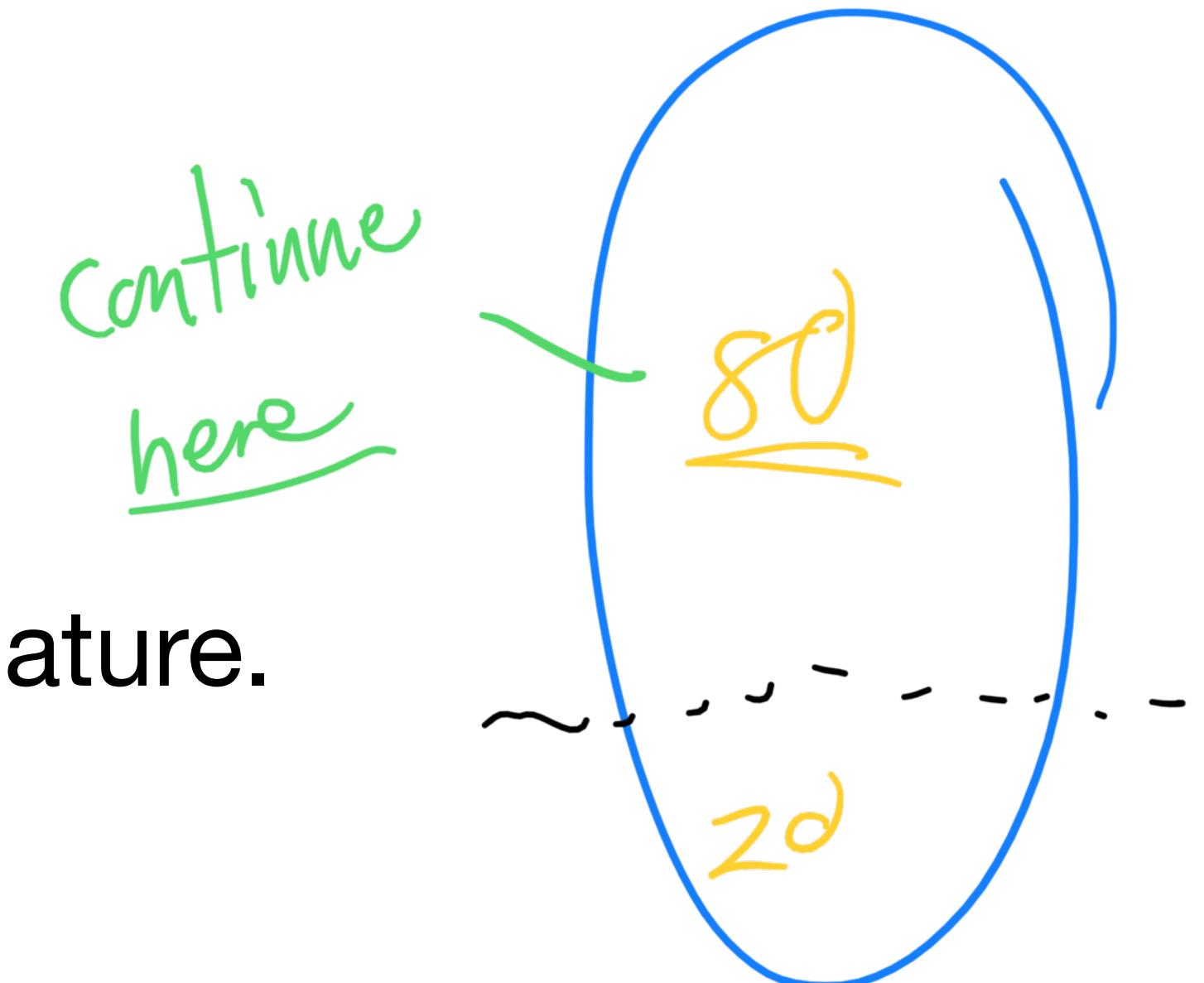
Idea: Send new block to nodes for validation and signature.

Correct nodes sign only one block at given depth.

Then collect certificate.

Certificate: If blocks require a certificate, we get similar properties.

- **Rate limit:**
- **Fork probability:**
- **Prevent system split:**
A subsystem, with few nodes cannot create certificates.



BFT protocol

Certificate vs. PoW problem

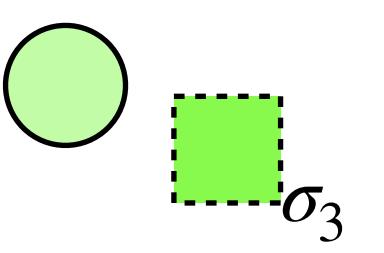
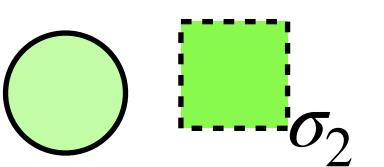
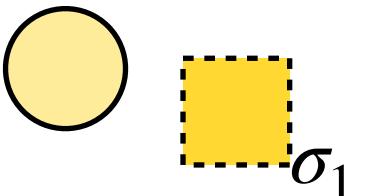
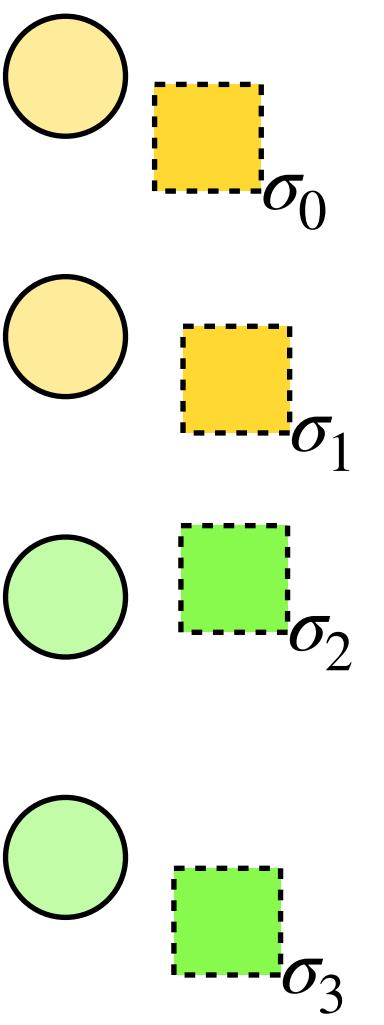
Idea: Send new block to nodes for validation and signature.

Correct nodes sign only one block at given depth.

Then collect certificate.

Problem: How to ensure that a certificate is created?

- Nodes may sign different blocks
- No block gets a certificate
- **Solution:**



nodes

BFT protocol

Certificate vs. PoW problem

Idea: Send new block to nodes for validation and signature.

Correct nodes sign only one block at given depth.

Then collect certificate.

Problem: How to ensure that a certificate is created?

- Nodes may sign different blocks
- No block gets a certificate
- **Solution:** Leader

(correct) leader

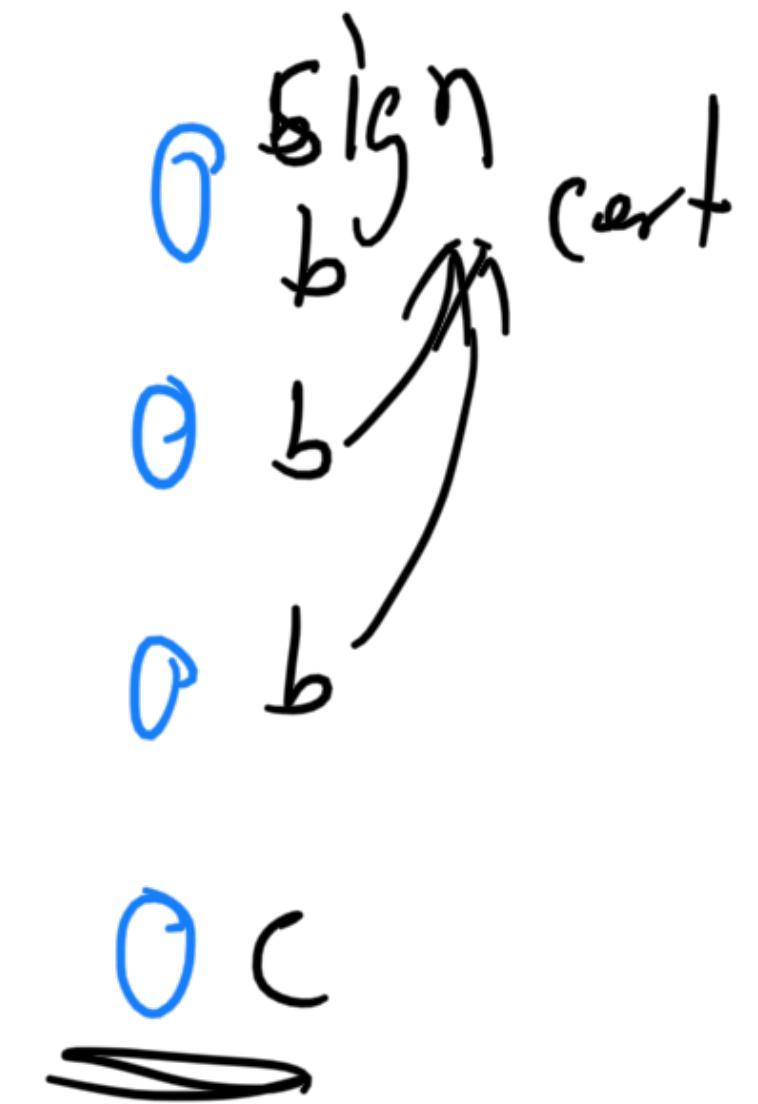
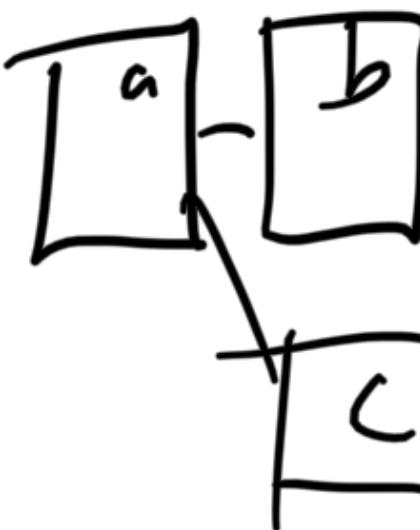
*→ propose only
one block*

faulty leader

→ change leader,

BFT protocol

Certificate vs. PoW problem



Idea: Send new block to nodes for validation and signature.
Correct nodes sign only one block at given depth.
Then collect certificate.

Problem: How to know that a certificate was created?

- A certificate may be collected by a single node
- The node with the certificate may fail and come back later
- **Solution:**

BFT protocol

Certificate vs. PoW problem

Idea: Send new block to nodes for validation and signature.

Correct nodes sign only one block at given depth.

Then collect certificate.

Problem: How to know that a certificate was created?

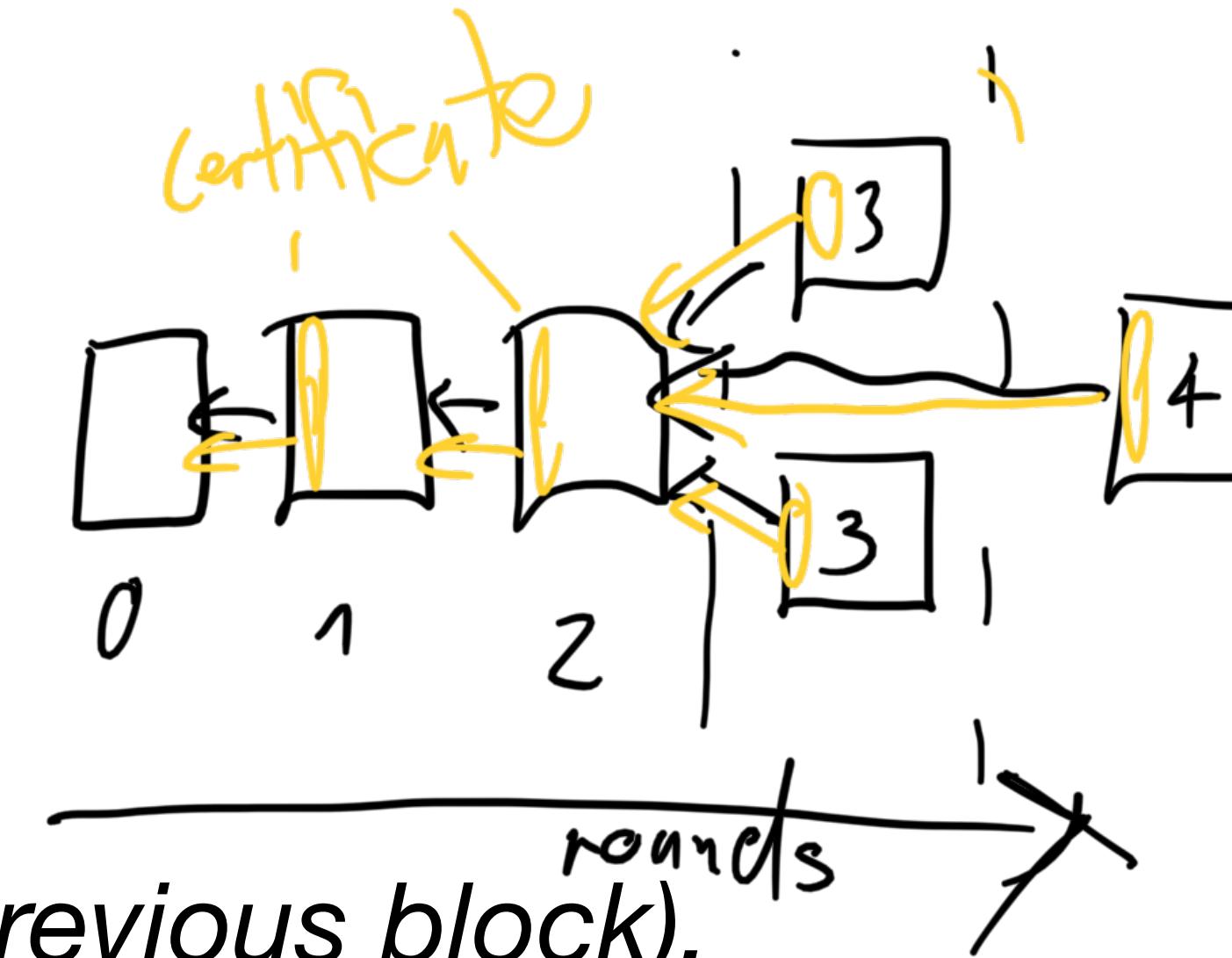
- A certificate may be collected by a single node
- The node with the certificate may fail and come back later
- **Solution:** Require multiple certificates

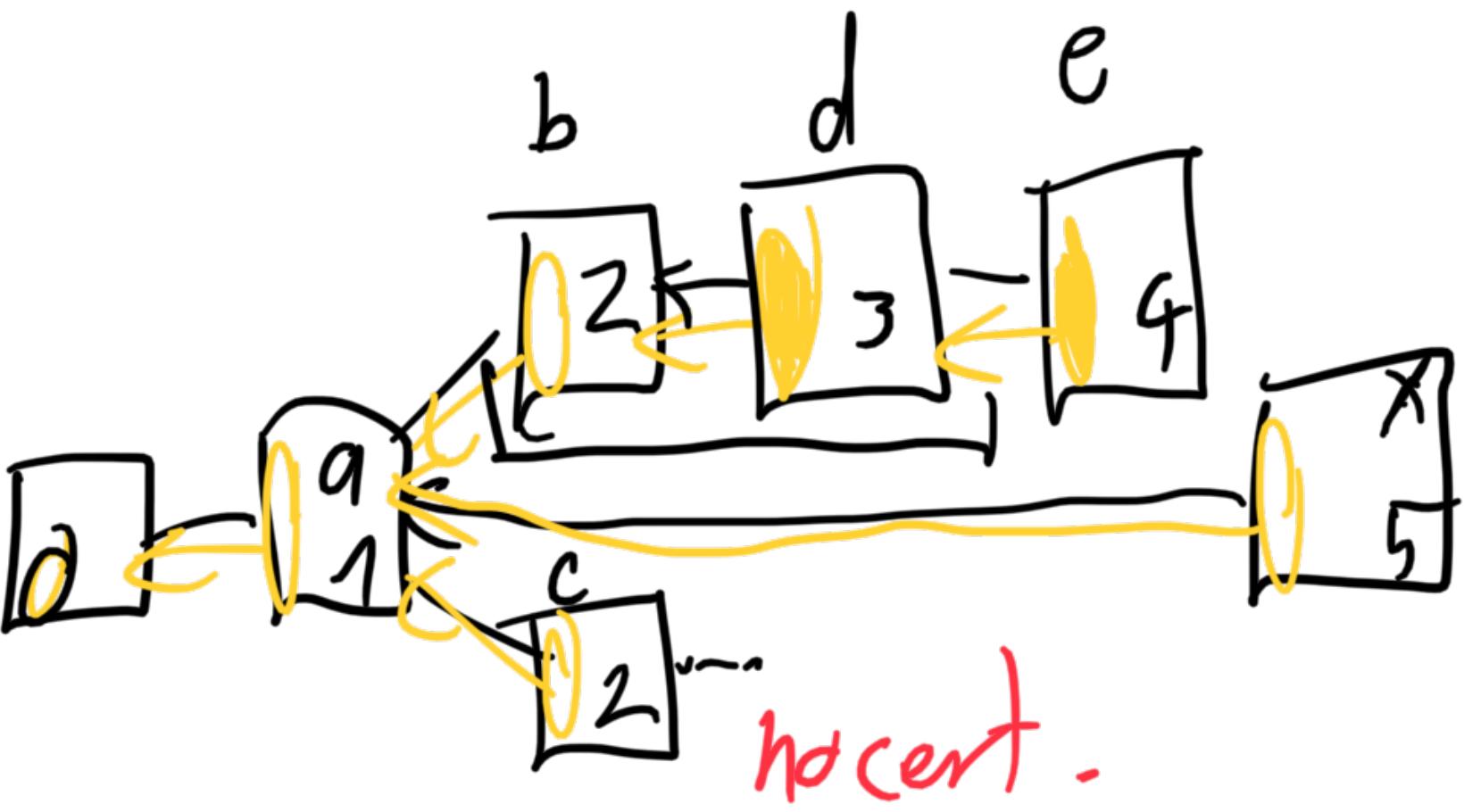
BFT protocol

Simple HotStuff (2 chain)

Preliminary:

- Every block includes a parent link (*previous block*).
=> Blocks form a **tree**.
Every block includes a **round number**
- Every block must include a **certificate** for its parent
- A block's **round** must be larger than that of its parent





max lock		
0	3	2
0	3	2
0	3	2
0		

- cannot sign X because of rank 2
- can sign e

BFT protocol

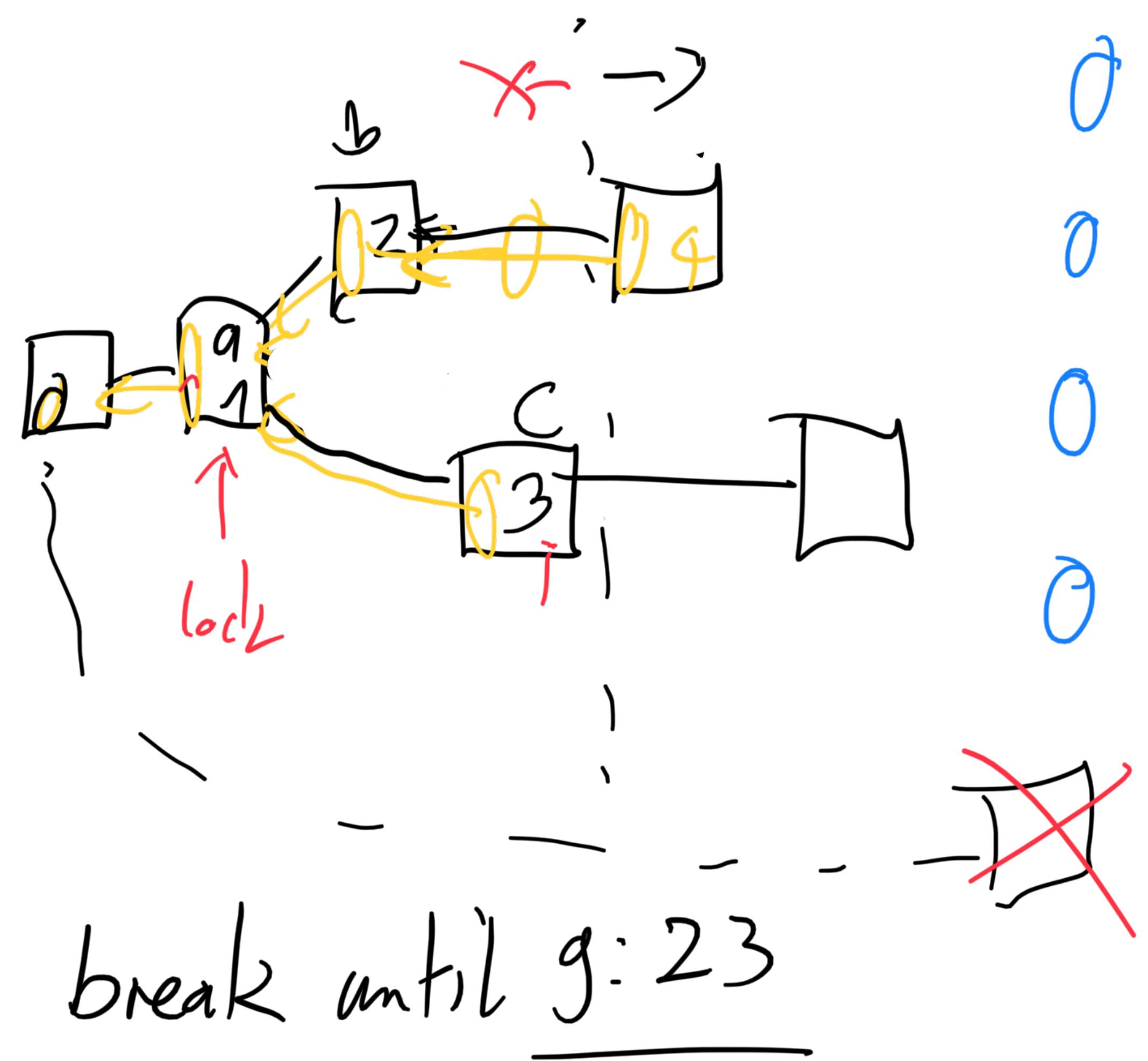
Simple HotStuff (2 chain)

Rules

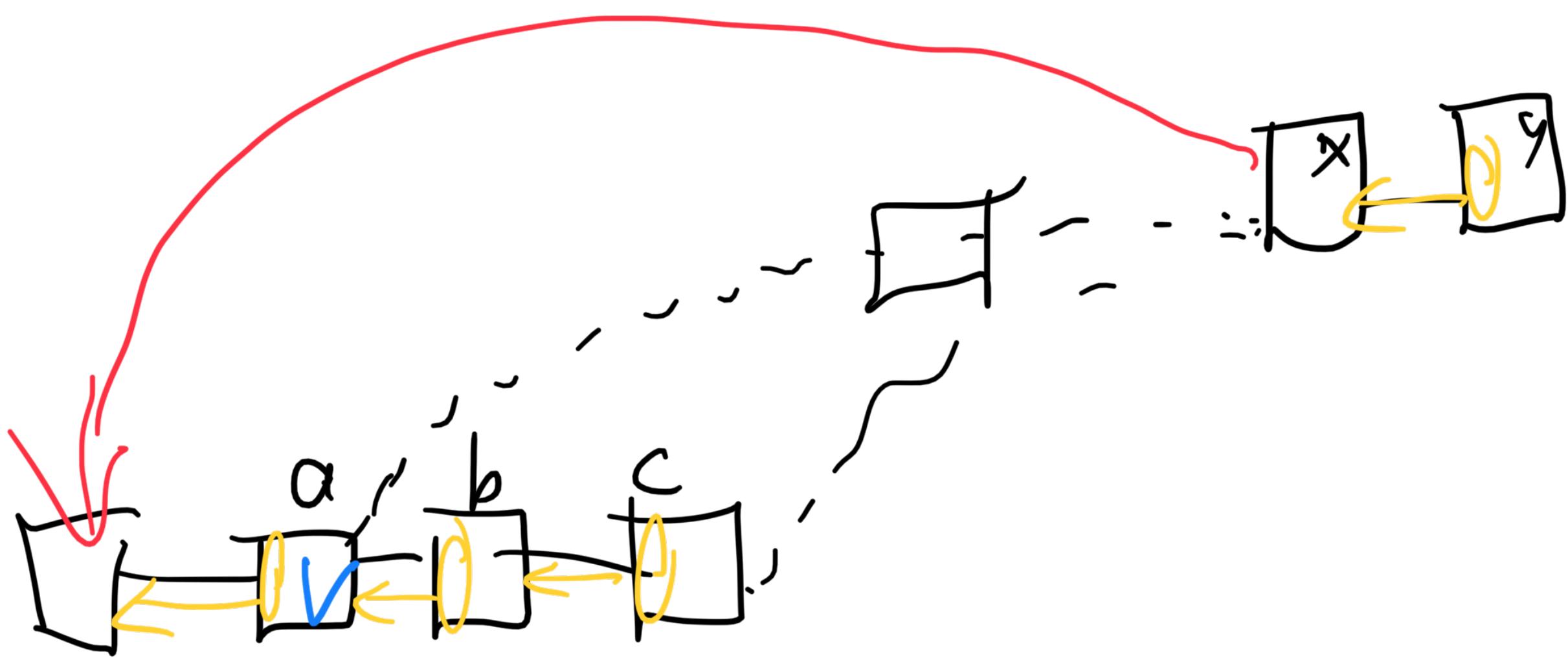
- Every block must contain certificate for parent
- Every block must have round > round of parent
- Rule 1: After signing a block at round $\max = r$,
a node may only sign at depth $r' > \max$.
- Rule 2: After signing a block with parent p and $p . round = lock$, only sign
blocks with parents in round $pr \geq lock$

Keep maximum value for \max and $lock$ in local variables.

no two blocks in
one round with certificate



	max	lock
0	1	0
0	2	1
0	2	1
0	2	1
0	2	1



BFT protocol

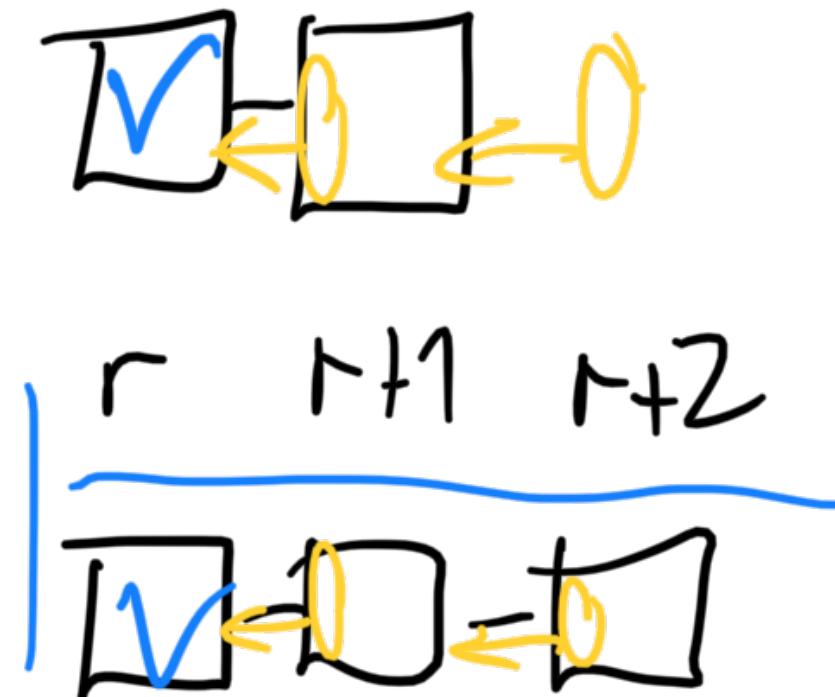
Simple HotStuff

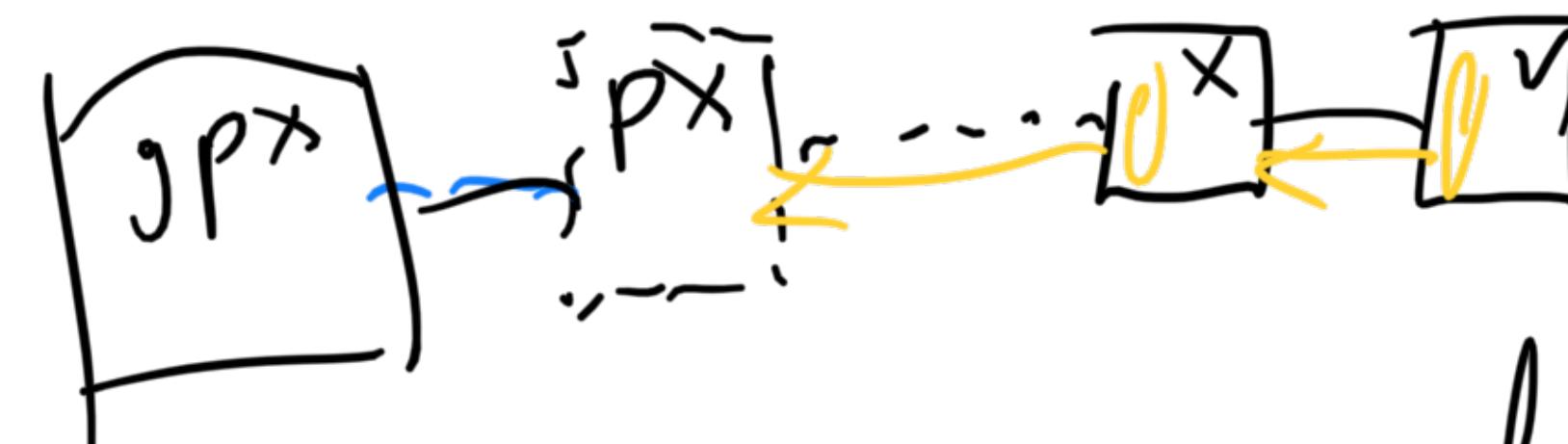
Def.:

- a) A block with $round = r$ is confirmed if it has a child in $round = r + 1$, which has a certificate.
- b) A block with $round = r$ is **confirmed** if it has a grandchild in $round = r + 2$

Theorem: If a block is confirmed, only descendants of that block, can get a certificate.

Proof: A majority of correct nodes have set their *lock* to the confirmed node.

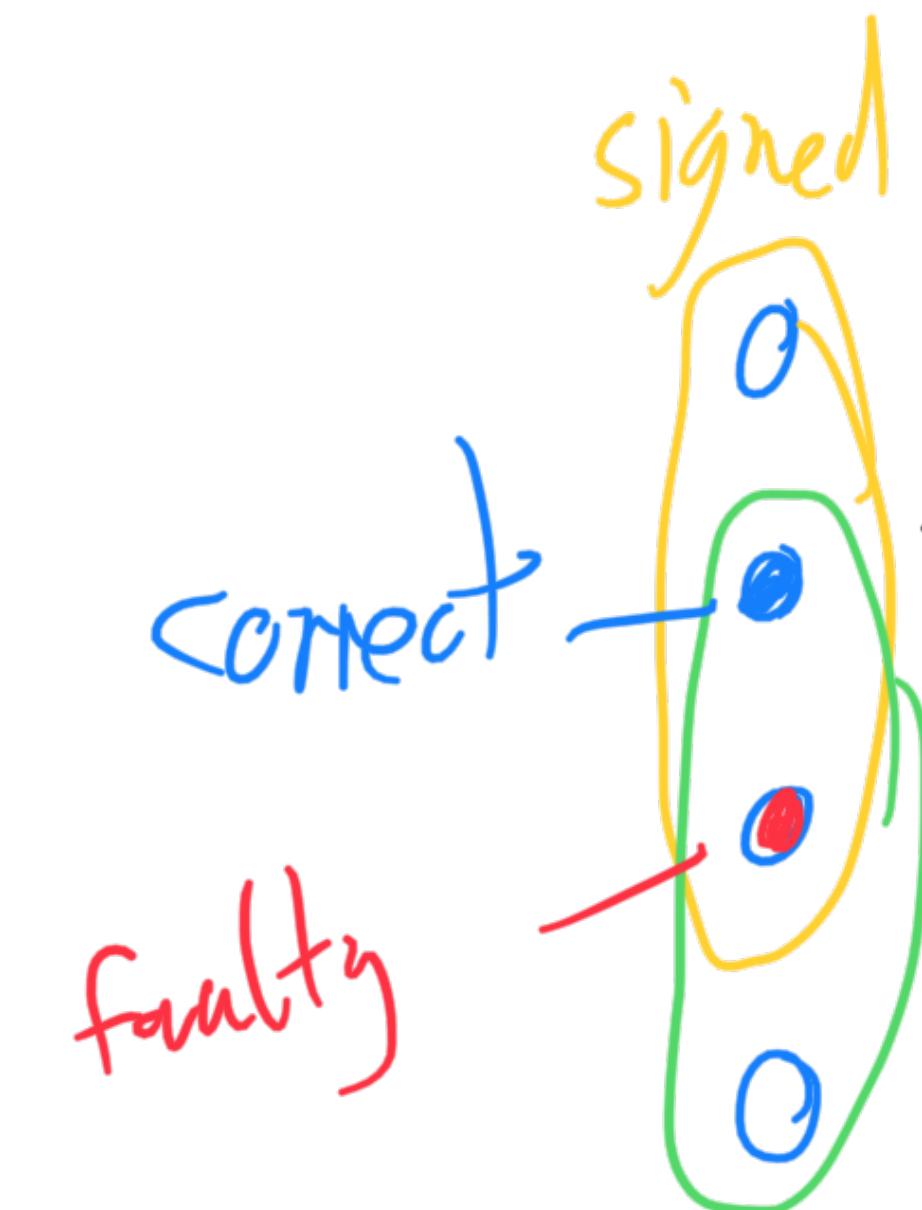




- round $x >$ round a



round $x - \text{round } a$



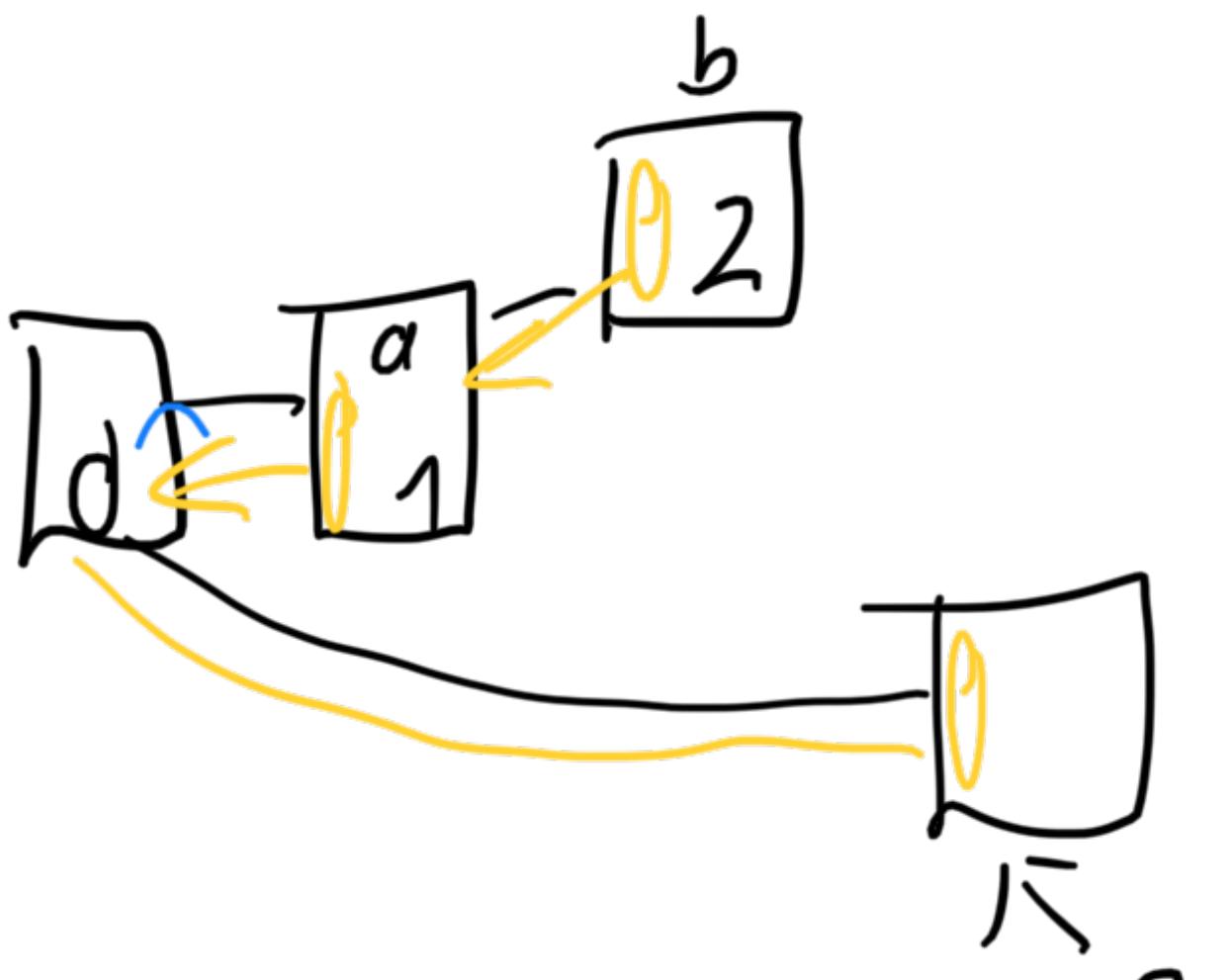
- rule1 \rightarrow round $x > \text{round } b$
- rule2 \rightarrow round $px \geq \text{round } a$
- d) round $px = \text{round } a$ ✓
 $\rightarrow px = a$
- signed \times b) round $px = \text{round } b$ ✓
 $\rightarrow px = b$
- c) round $px > \text{round } b$

BFT protocol

Simple HotStuff - Leader

Idea 1: Every round has a designated leader.

Idea 2: Nodes wait for Δ time for a proposal in current round, before accepting at next round.



P_1/P_2 cannot sign

~~ask for last cert.~~

	max	lock lastcert
P_1	0	2
P_2	0	2
P_3	0	1
P_4	0	0

BFT protocol

Simple HotStuff - Leader

Idea 1: Every round has designated leader.

Idea 2: Nodes wait for Δ time for a proposal in current round, before accepting at next round.

How can a leader avoid the situation from the example?

Ask all nodes for most recent certificate.

Wait for Δ time to receive proposal from all correct nodes.

BFT protocol

Simple HotStuff (3 chain)

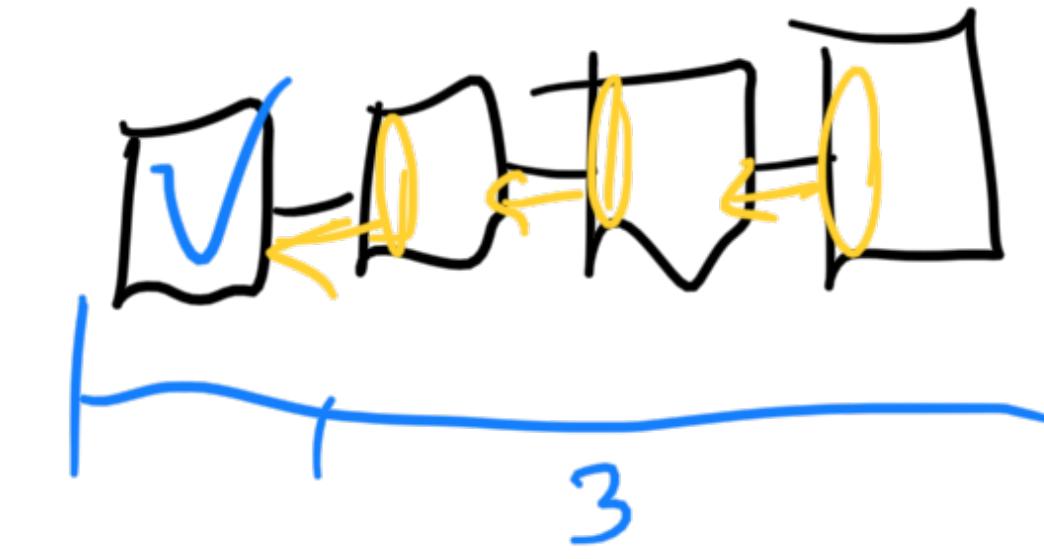
Rules

- *Every block must contain certificate for parent*
- *Every block must have round > round of parent*
- **Rule 1:** After signing a block at round $\max = r$,
a node may only sign at depth $r' > \max$.
- **Rule 2:** After signing a block with grandparent p and $p . round = lock$, only sign
blocks with parents in round $pr \geq lock$

Keep maximum value for \max and $lock$ in local variables.

BFT protocol

Simple HotStuff (3chain)



Def.:

- a) A block with $round = r$ is **confirmed** if it has a grandchild in $round = \underline{r + 2}$, which has a certificate.
- b) A block with round = r is **confirmed** if it has a grand-grandchild in $\underline{round = r + 3}$

Theorem: If a block is confirmed, only descendants of that block, can get a certificate.

Proof: A majority of correct nodes have set their *lock* to the confirmed node.

BFT protocol

Simple HotStuff - Leader

Idea 1: Every round has designated leader.

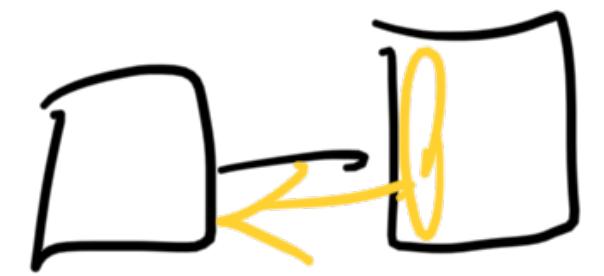
Idea 2: Nodes wait for Δ time for a proposal in current round, before accepting at next round.

How can a leader avoid the situation from the example?

Ask all nodes for most recent certificate.

Wait for $2f + 1$ replies





- Certificates instead of PoW
 - fixed membership
- Problem of misbehavior
 - need more than majority
- Problem of deadlock
 - leader
 - confirmed after multiple blocks.