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

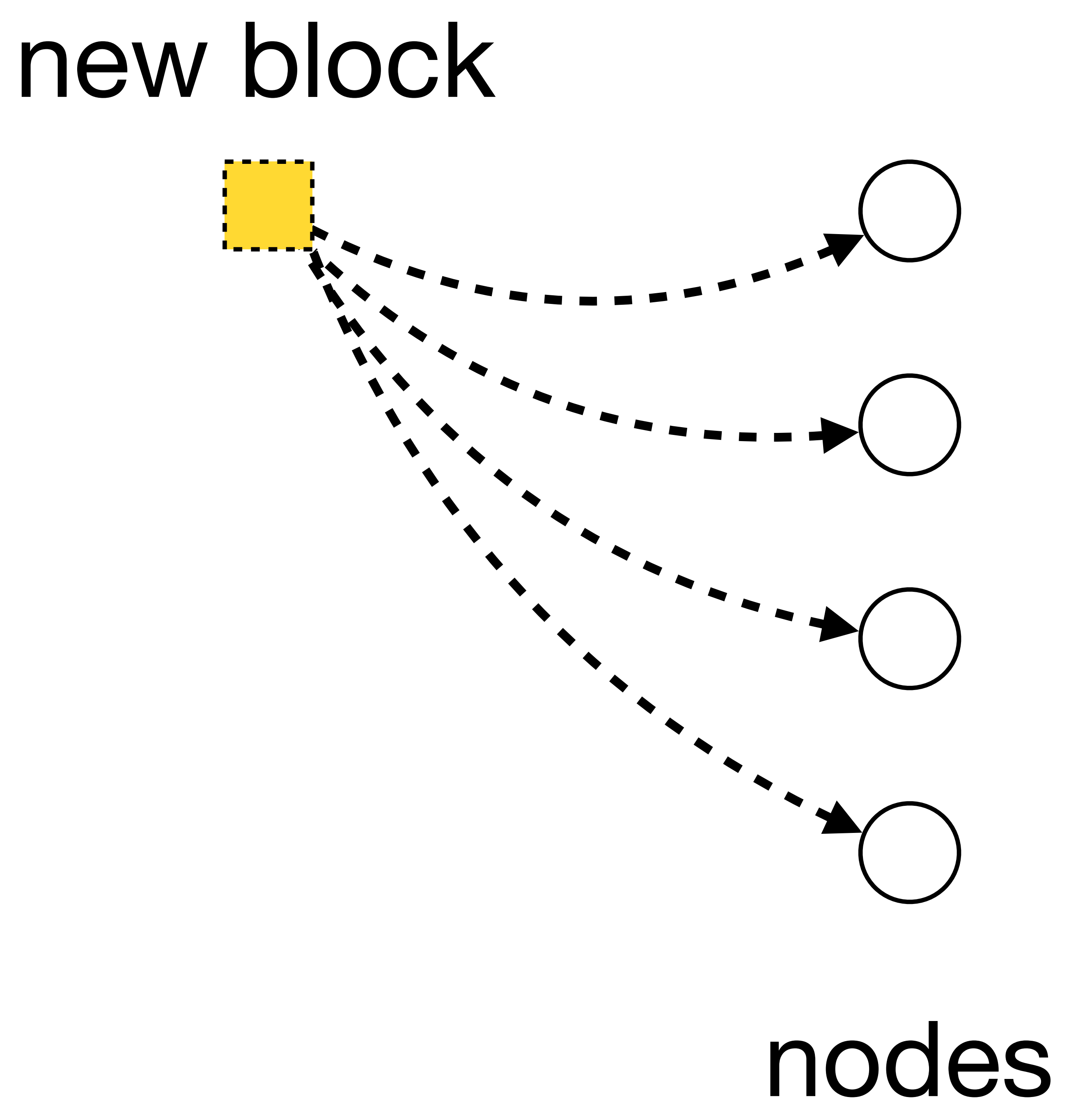
Certificate: If blocks require a certificate, we get similar properties.

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

BFT protocol

Certificate vs. PoW

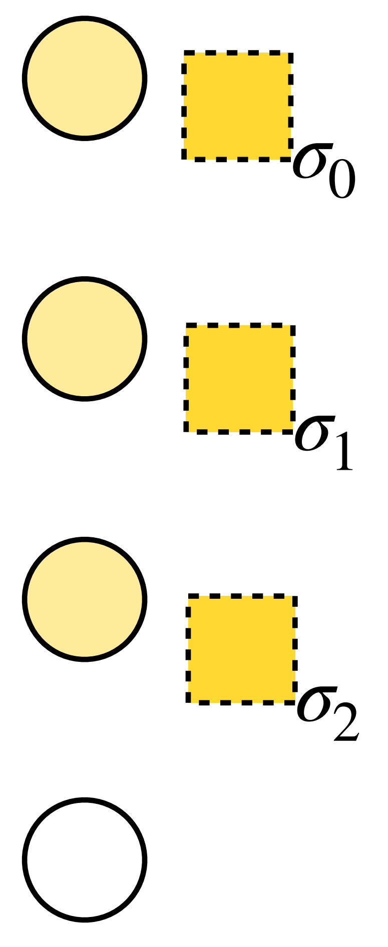
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

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

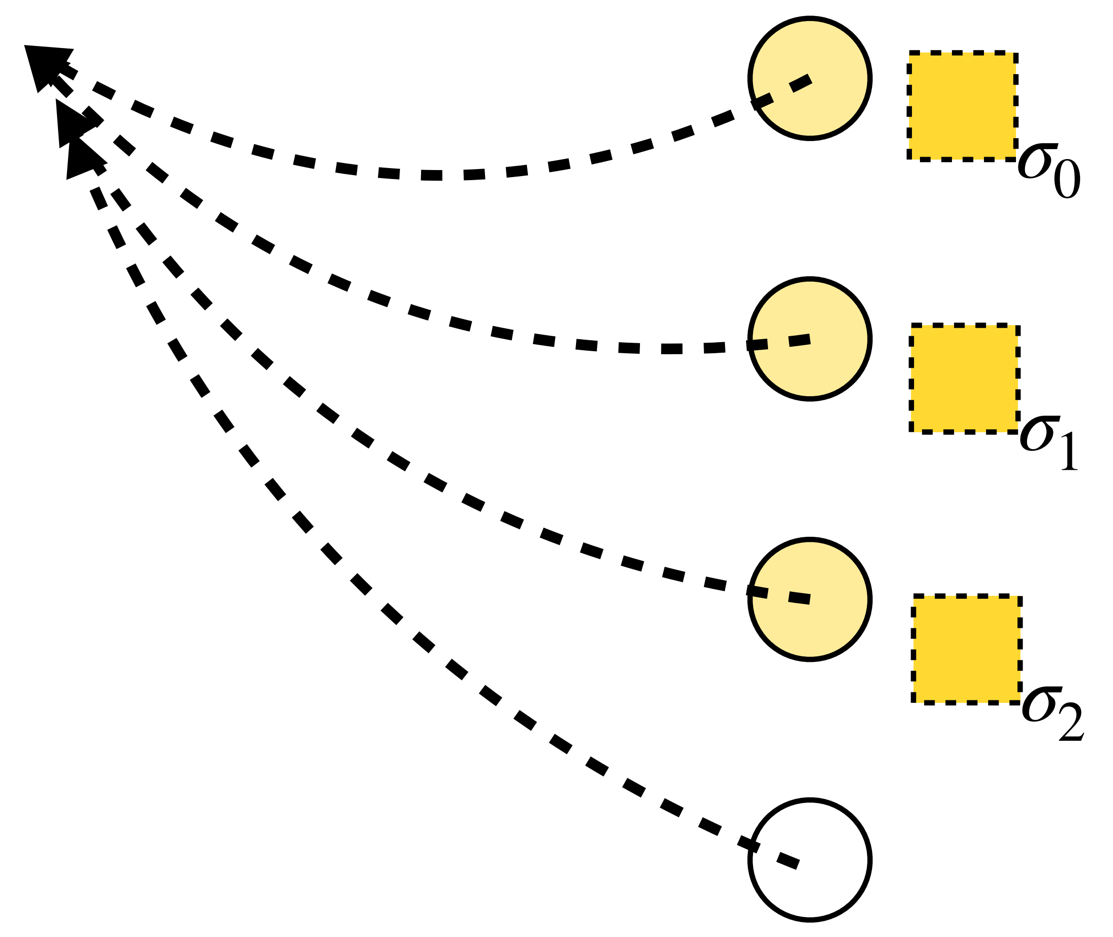


nodes

BFT protocol

Certificate vs. PoW

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

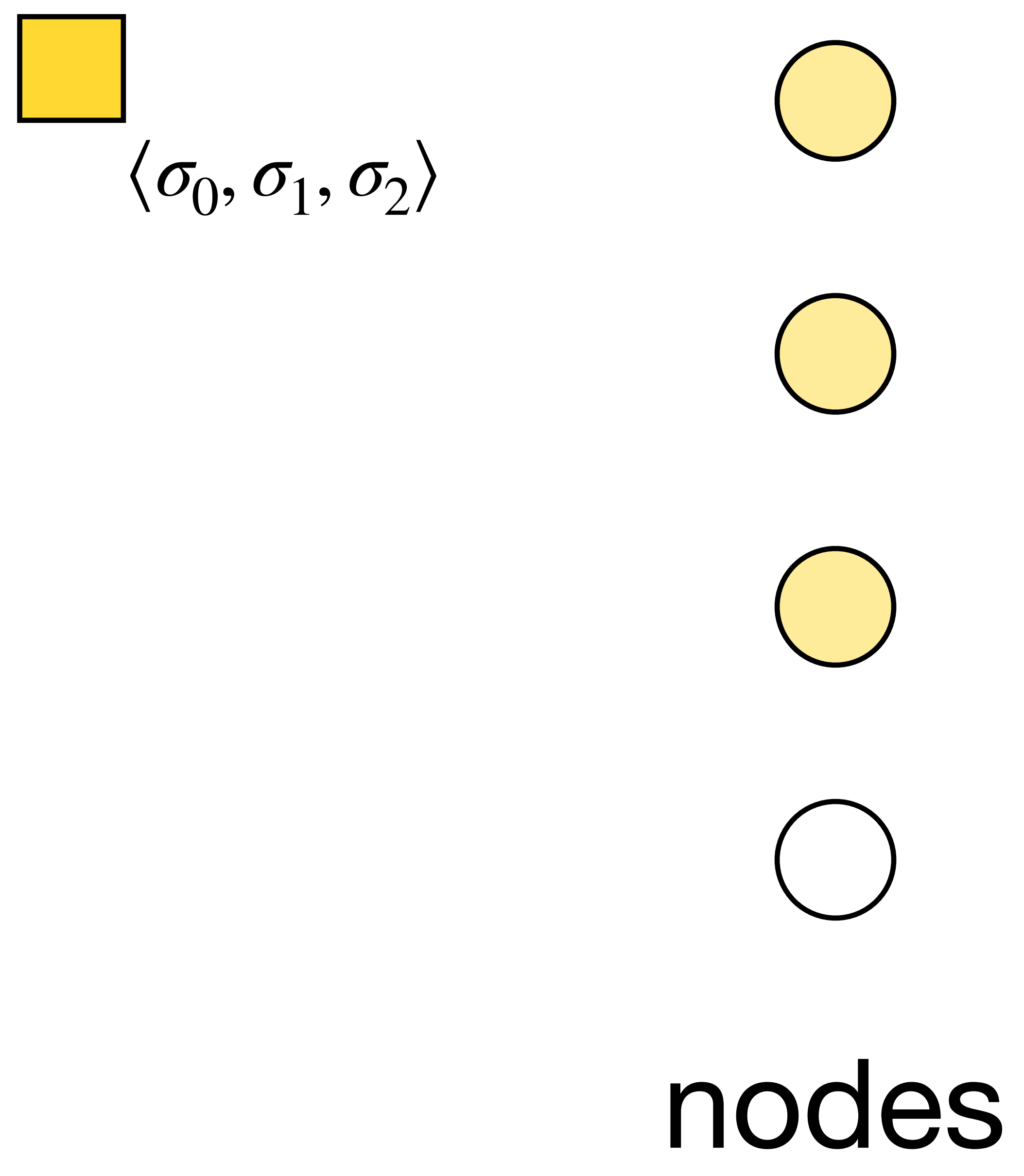


nodes

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.

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:**

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:**

BFT protocol

Model

Model:

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

Certificate:

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

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.

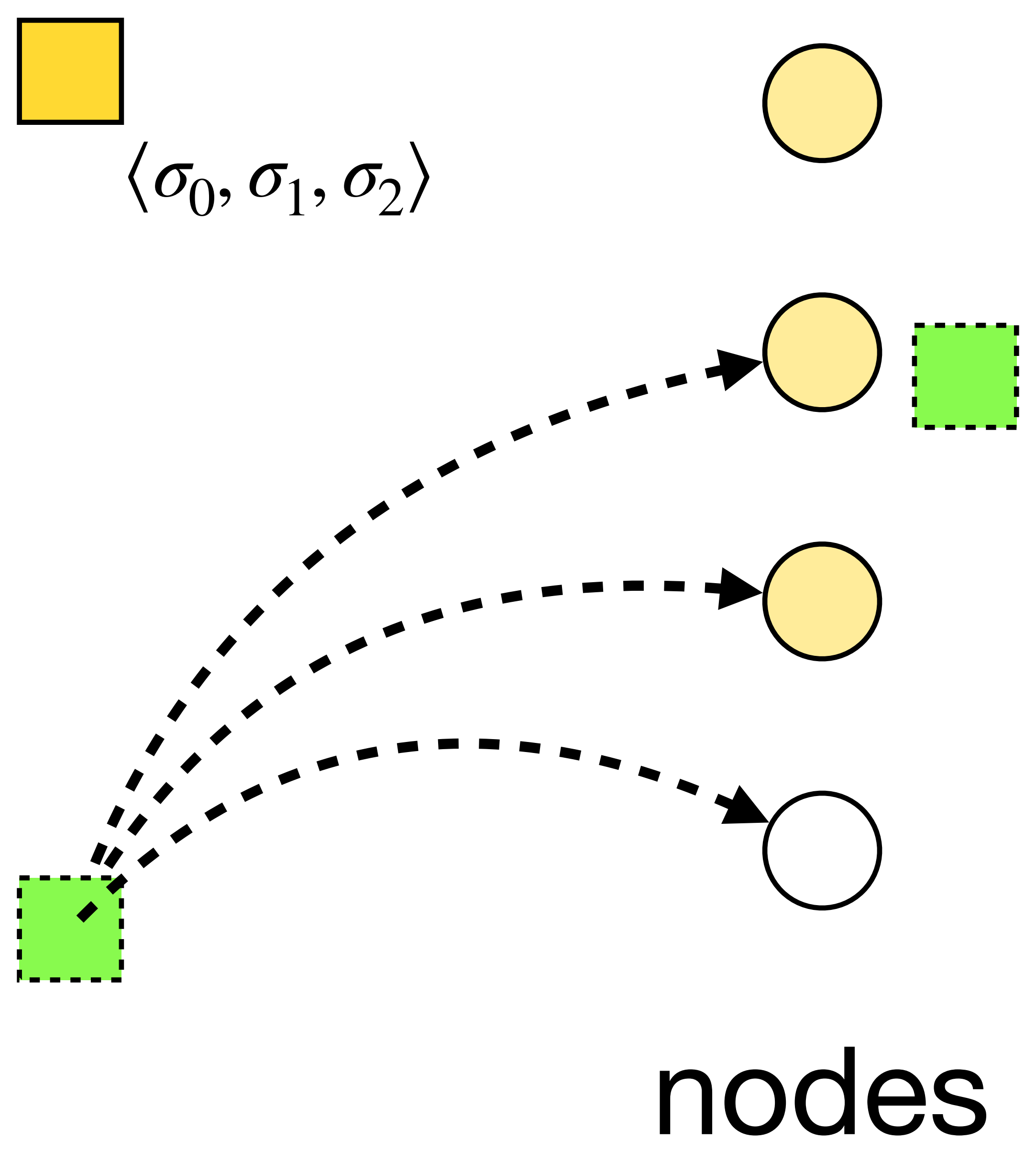
Certificate: If blocks require a certificate, we get similar properties.

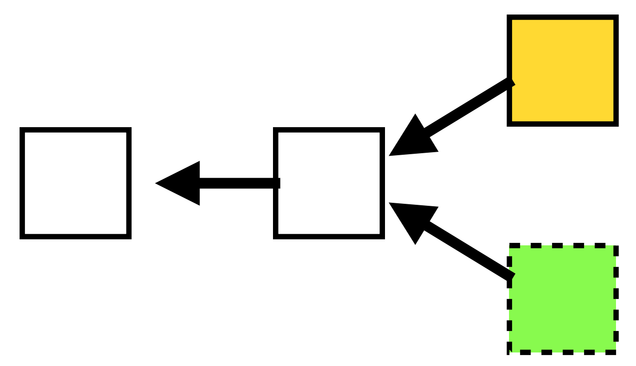
- **Rate limit:**
- **Fork probability:**
  If nodes do not sign multiple blocks, at most one block at a given height can get a certificate.
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

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

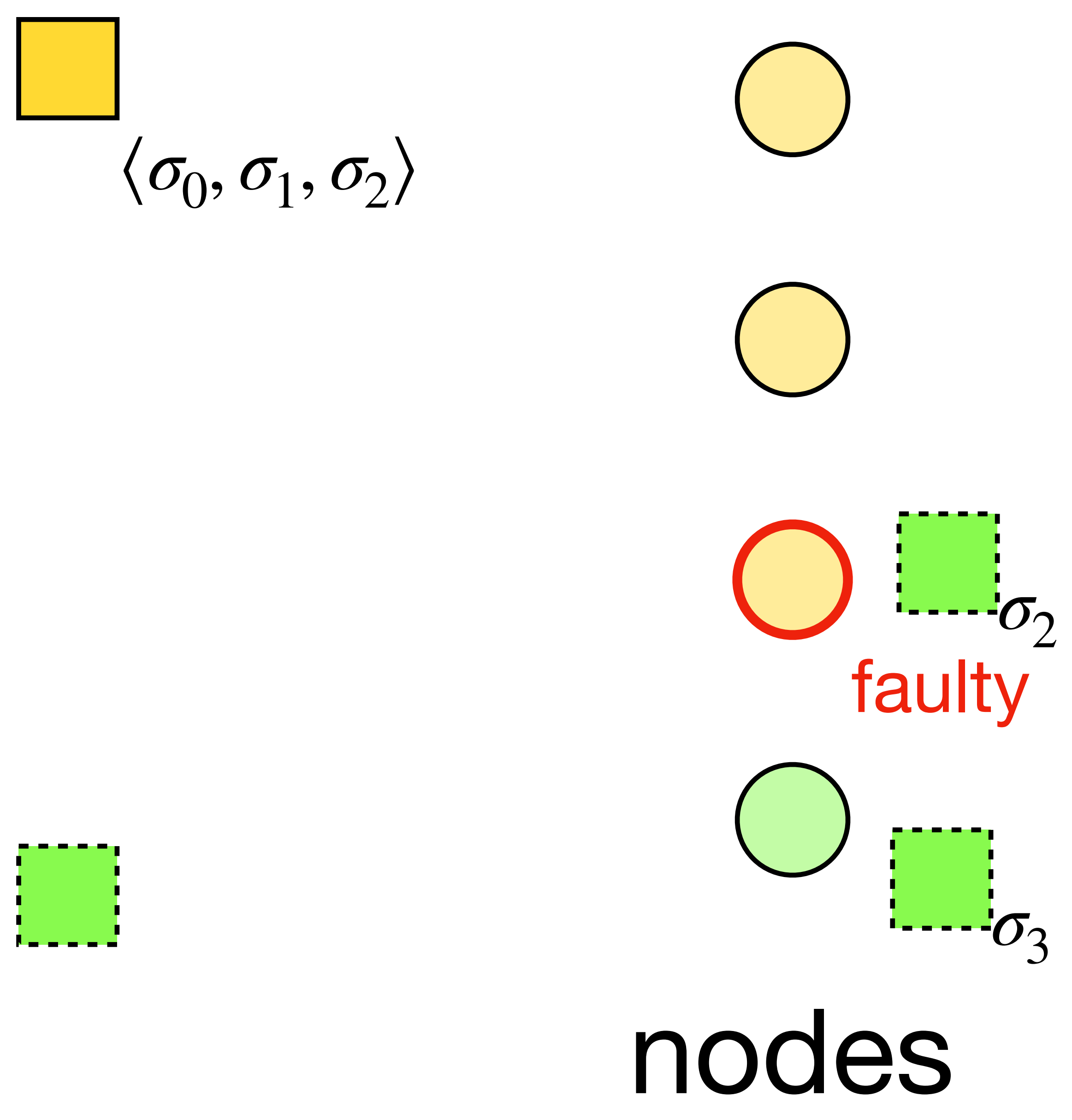


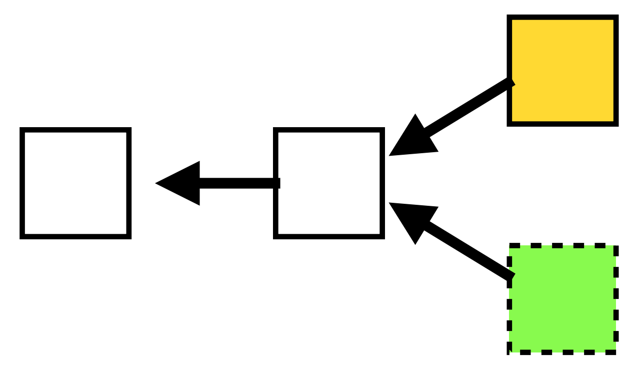


BFT protocol

Certificate vs. PoW

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

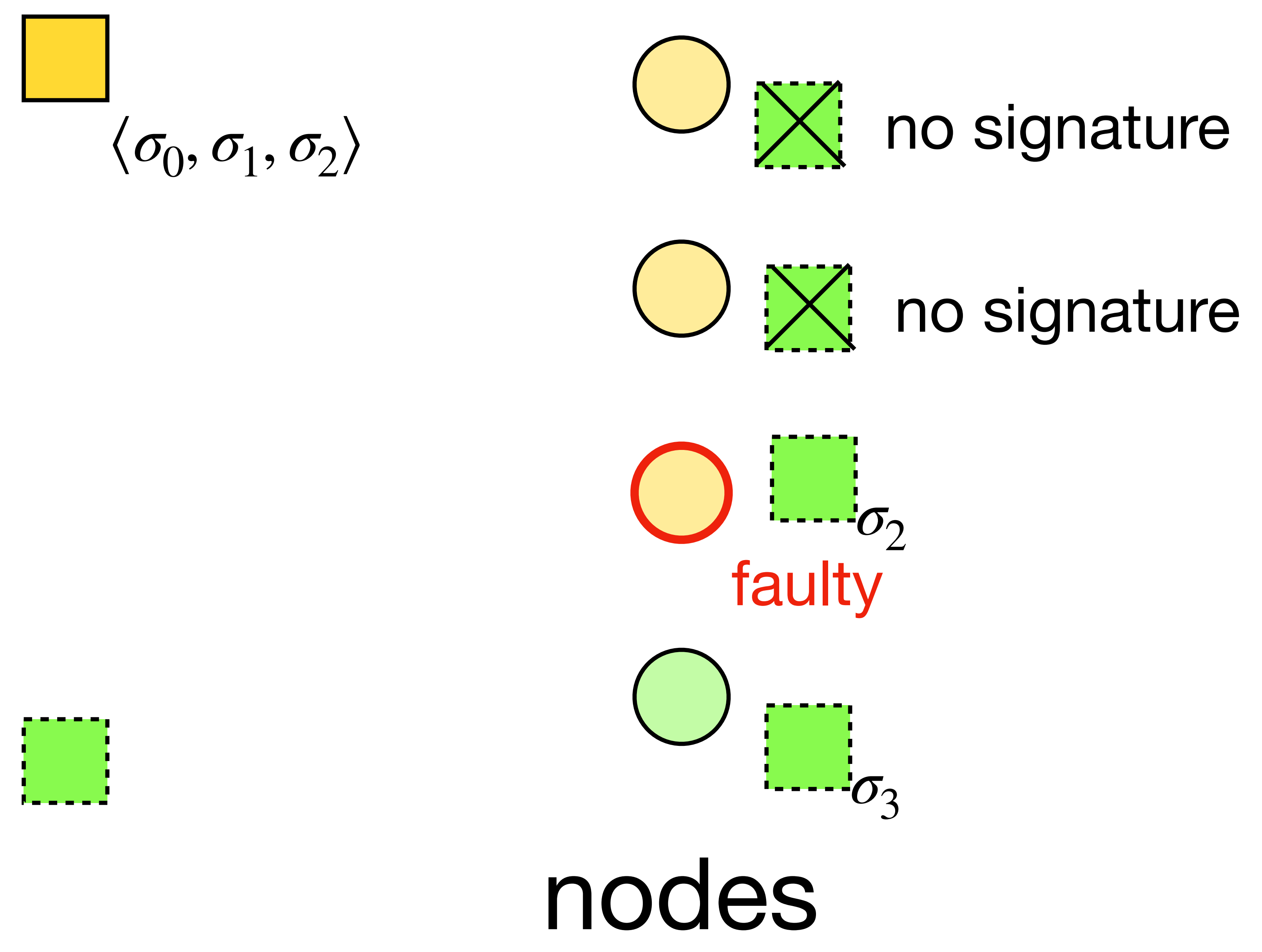


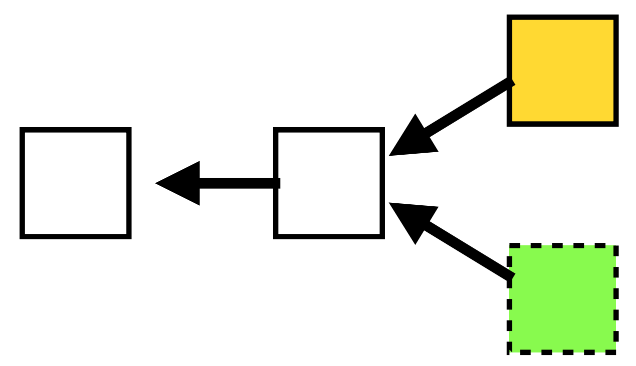


BFT protocol

Certificate vs. PoW

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

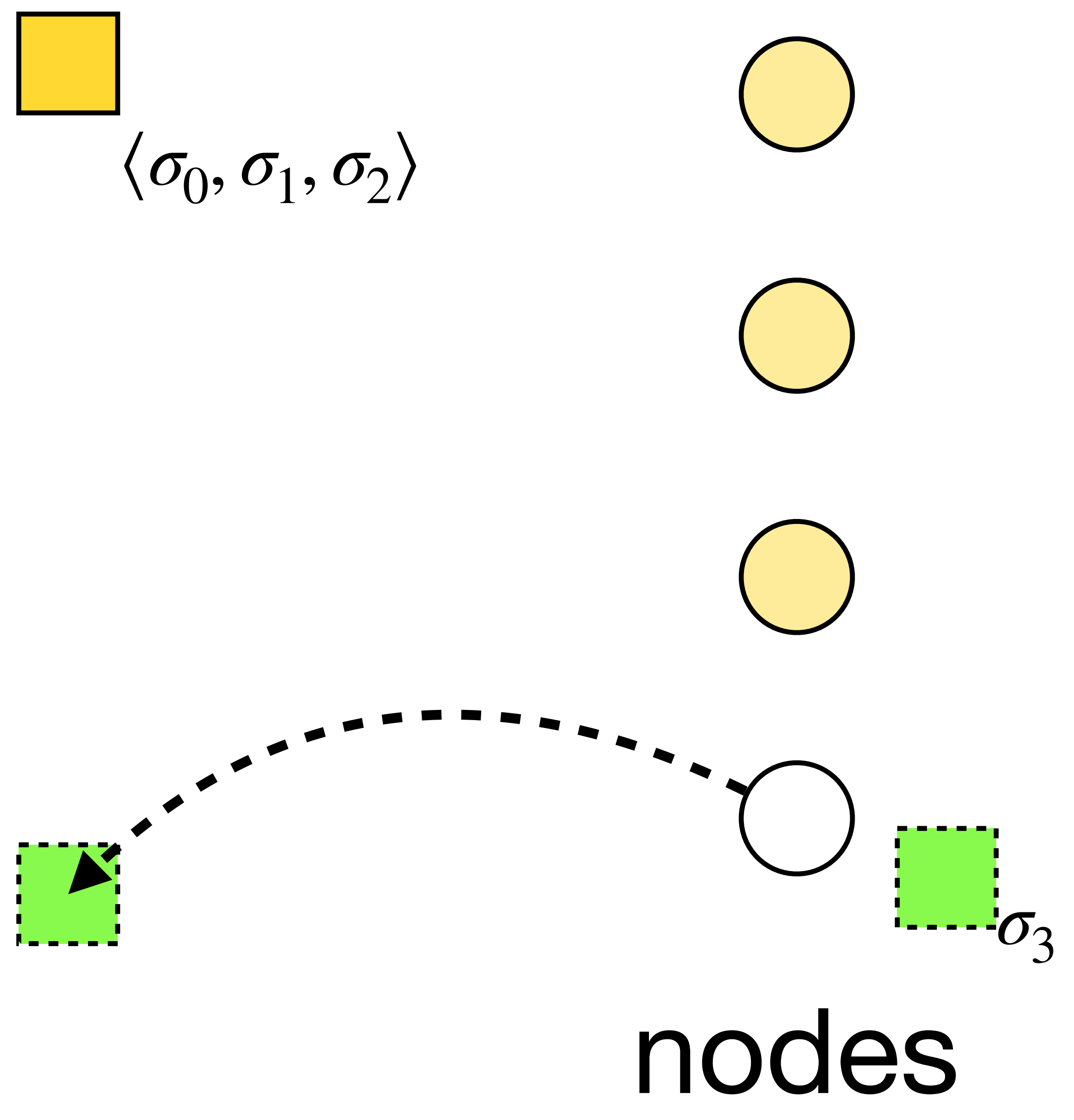


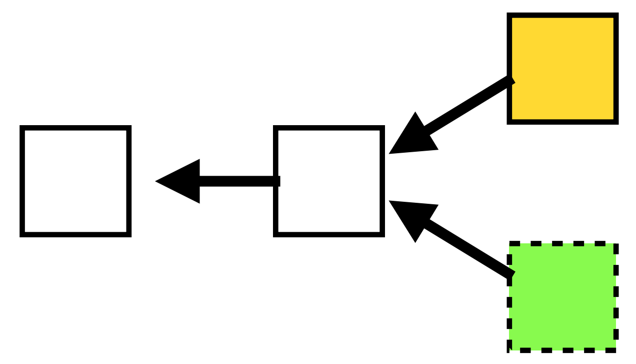


BFT protocol

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

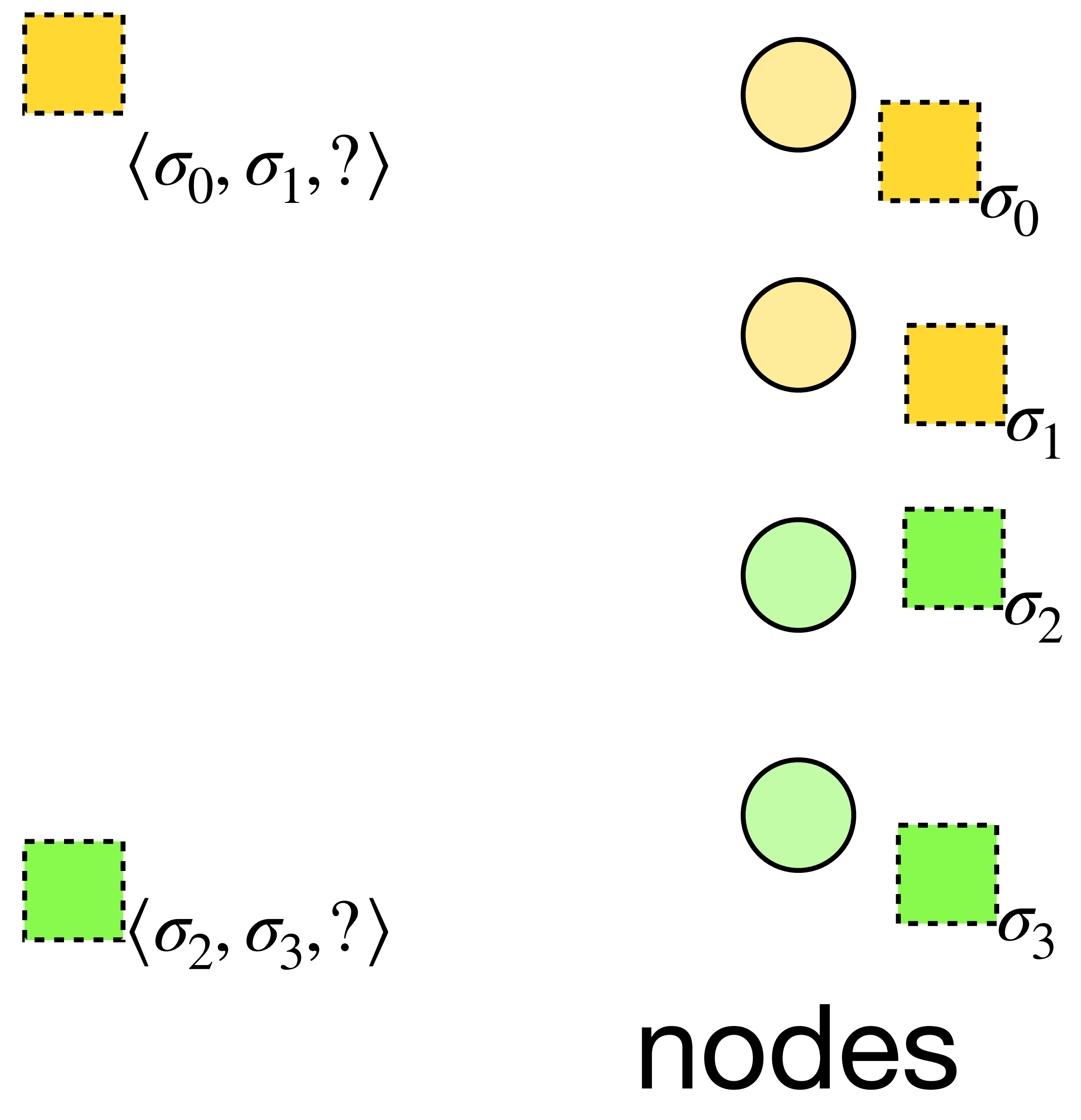
Certificate vs. PoW problem

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:**



$\langle \sigma_0, \sigma_1, ? \rangle$
$\langle \sigma_2, \sigma_3, ? \rangle$
σ_0
σ_1
σ_2
σ_3
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

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 blocks **round** must be larger than that of its parent

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 round $r' > max$.
- **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.

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.

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.

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 round $r' > max$.
- **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)

In this variant we need to wayt longer for confirmation!

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

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

A no leader needs not wait for Δ time!