A BFT protocol

Simplified HotStuff

Leander Jehl

BFT protocol

Model

Model:

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

Certificate:

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

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

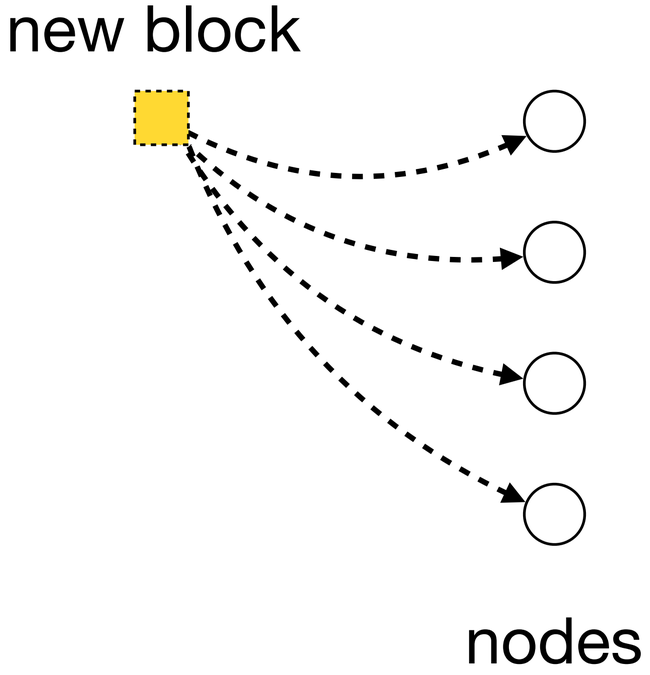
Certificate: If blocks require a certificate, we get similar properties.

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

BFT protocol

Certificate vs. PoW

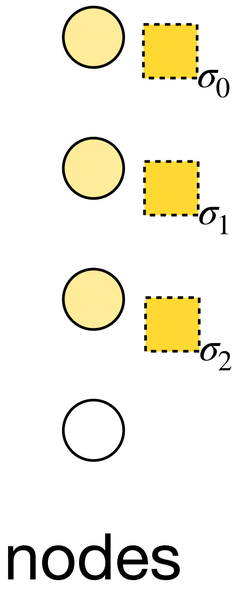
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

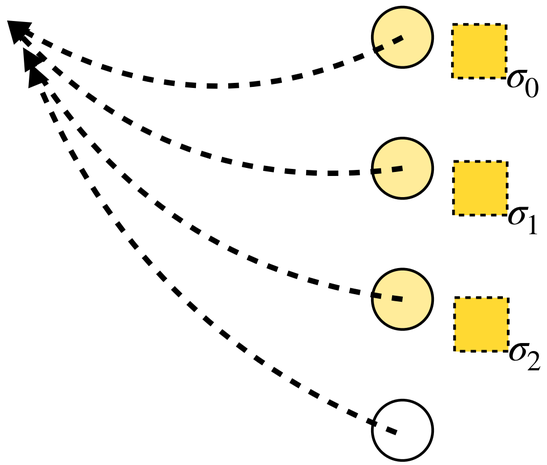
Idea: Send new block to nodes for validation and signature.
Then collect certificate.



BFT protocol

Certificate vs. PoW

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

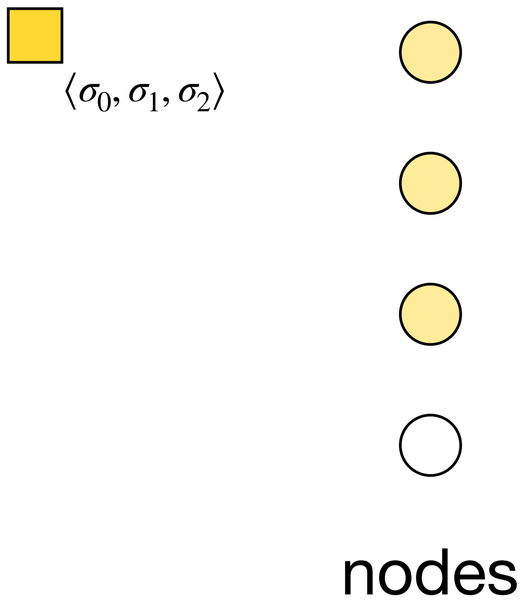


nodes

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.

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.

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

- **Rate limit:**

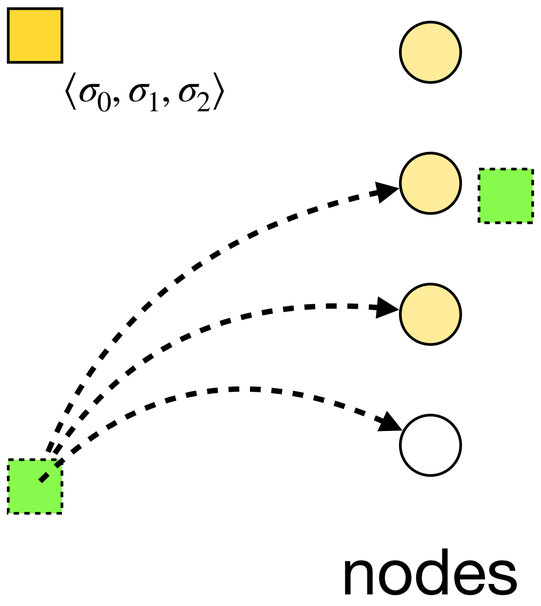
- **Fork probability:**
  If nodes do not sign multiple blocks, at most one block at a given height can get a certificate.

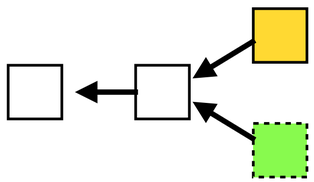
- **Prevent system split:**

BFT protocol

Certificate vs. PoW

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

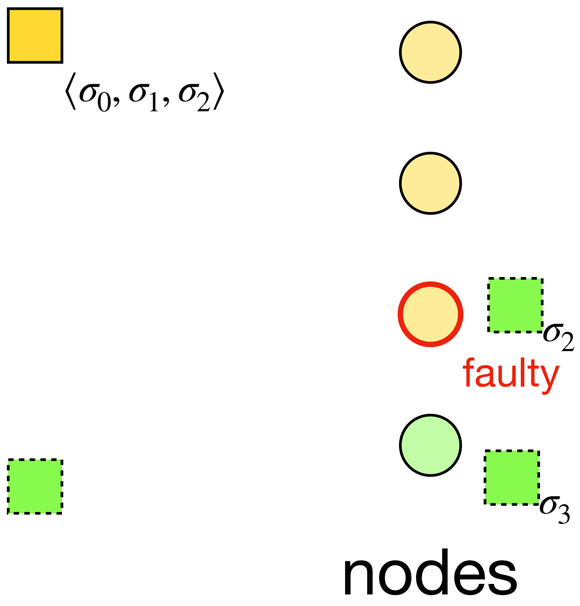


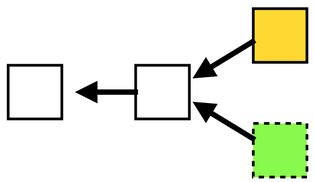


BFT protocol

Certificate vs. PoW

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

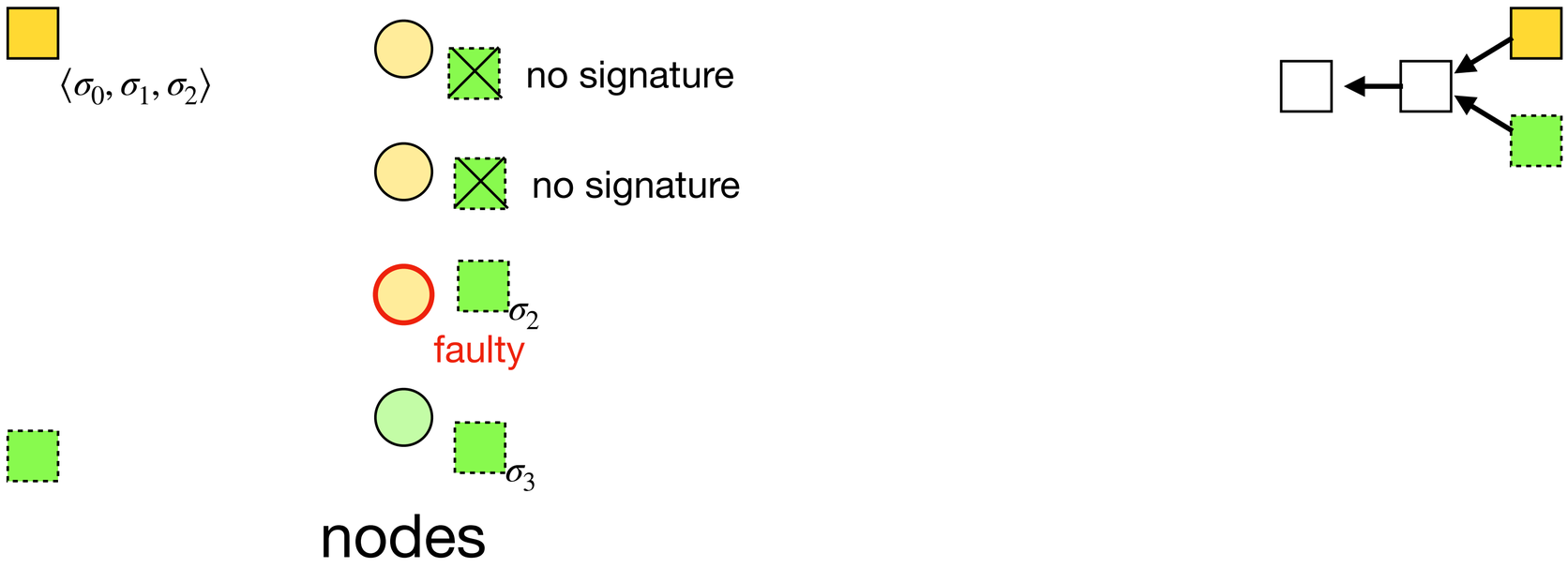




BFT protocol

Certificate vs. PoW

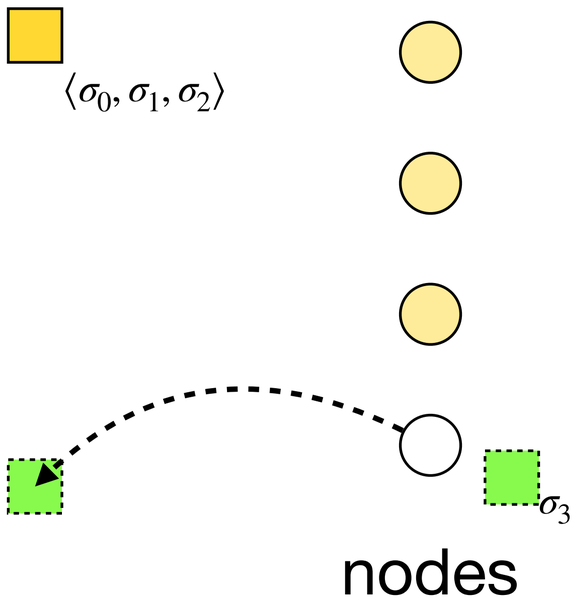
Idea: Send new block to nodes for validation and signature.
Then collect certificate.

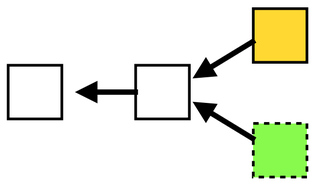


BFT protocol

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

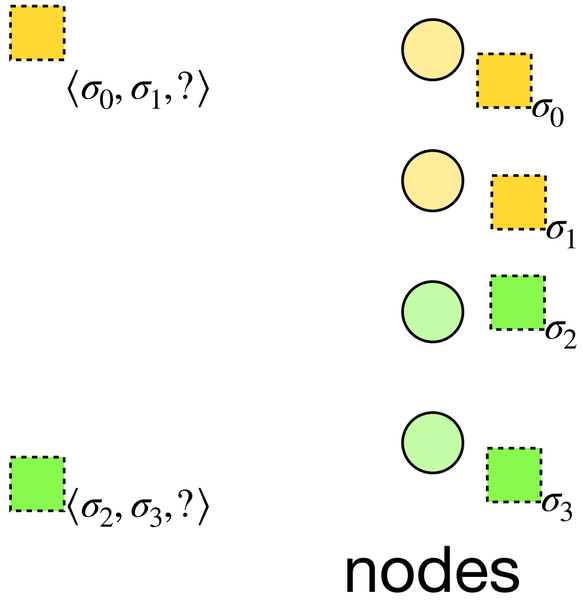
Certificate vs. PoW problem

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:**



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.**
  A blocks *depth* is the distance from the root (genesis block).

- Every block may includes a **certificate** for an ancestor.

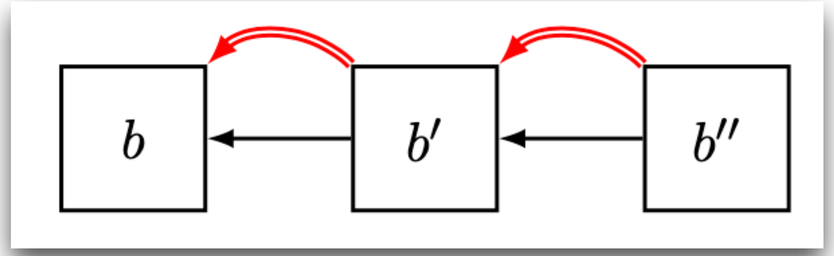
BFT protocol

Simple HotStuff (2 chain)

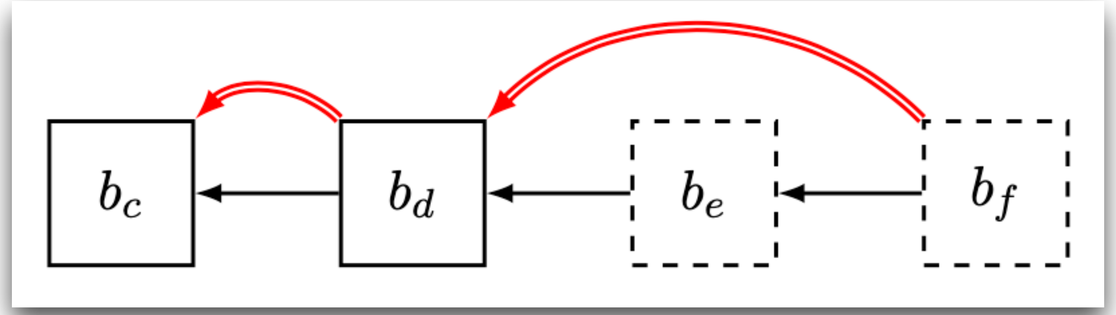
Preliminary:

- Every block includes a parent link (*previous block).*
  => Blocks form a **tree.**
  A blocks *depth* is the distance from the root (genesis block).

- Every block may includes a **certificate** for an ancestor.
  This is called **justification**.



Blocks with certificate for parent.



Blocks with certificates.

BFT protocol

Simple HotStuff (2 chain)

Rules

- **Rule 1:** After signing a block as depth d,
  a node may only sign at depth $d' > d$.

Every node maintains the **locked block,** i.e. the block at largest hight for which it has seen a certificate.

- **Rule 2:** A node only signs a block, if it is a decendant of the locked block.

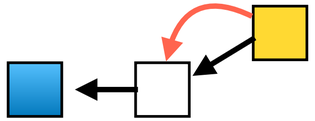
Obs: a node may update the locked block, based on the certificate included in a block.

BFT protocol

Example

Example

- Nodes n_0, n_1, and n_2 sign block
- They set $lock$ to

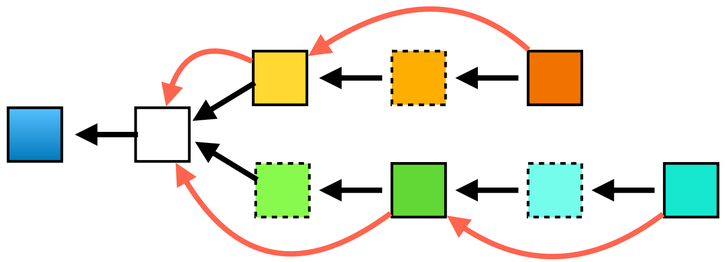


BFT protocol

Example

Example (bad case)

- n_0, n_1, and n_2 sign block
- n_1, and n_2 set $lock$ to
- n_3 creates
- n_1, and n_2 set $lock$ to
- n_3, n_1, and n_2 sign block

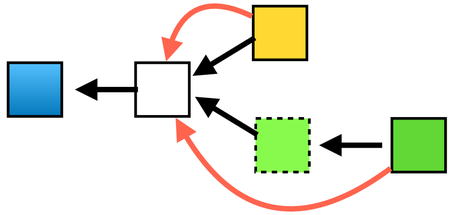


BFT protocol

Example

Example (recall)

- Nodes n_0, n_1, and n_2 sign block
- They set $lock$ to
- n_3 signs
- n_3 creates
- n_3, n_1, and n_2 sign block

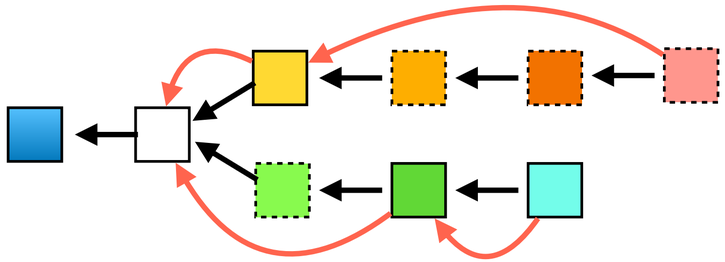


BFT protocol

Example

Example (good case)

- n_3 creates
- n_1, and n_2 set $lock$ to
- n_3, n_1, and n_2 sign block
- n_0 creates
- n_1, and n_2 will not sign

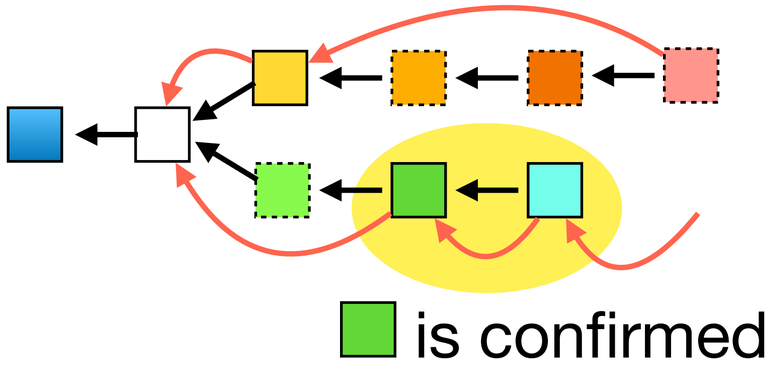


BFT protocol

Example

Example (good case)

- n_3 creates
- n_1, and n_2 set $lock$ to
- n_3, n_1, and n_2 sign block
- n_0 creates
- n_1, and n_2 will not sign



BFT protocol

Simple HotStuff

Def.: A block is **confirmed** if both the block and it's successor have a certificate.

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 depth has designated leader.

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

BFT protocol

Simple HotStuff - Leader

Idea 1: Every depth has designated leader.

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

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.