BFT and Hybrid solutions

HotStuff, ByzCoin, CasperFFG, and Algorand

Leander Jehl

HotStuff BFT

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.

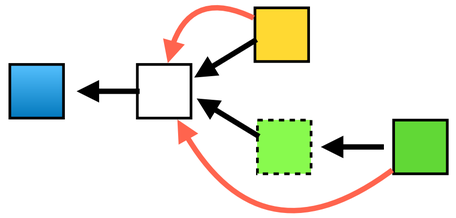
- **Rule 2:** A node only signs a block, if it is a decendant of the locked block.

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

Simple HotStuff
Example

Example

- 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



Node	$lock$
n_0	
n_1	faulty
n_2	
n_3	

Simple HotStuff

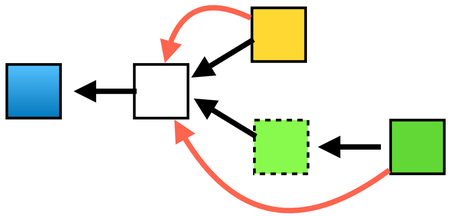
Example

Example

- assume node n_0 is new leader.
- to get a certificate, n_0 must either
  - extend ■, or
  - rely on faulty n_1

Solution:

n_0 waits for Δ time to get newest block



Node	$lock$
n_0	■
n_1	faulty
n_2	□
n_3	■

HotStuff (3 chain)

Rules

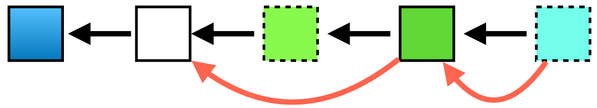
- **Rule 1:** After signing a block as depth d,
  a node may only sign at depth $d' > d$.

Every node maintains the $lock_3$ **block,** i.e. the block at largest hight, such that this block and one successor have a certificate.

- **Rule 2:** A node only signs a block b, if it is a decendant of the $lock_3$ block,
  or if some ancestor b' of b has a certificate, and b' has higher depth than $lock_3$

HotStuff (3 chain)

Example $lock_3$

- **Example 1:** 

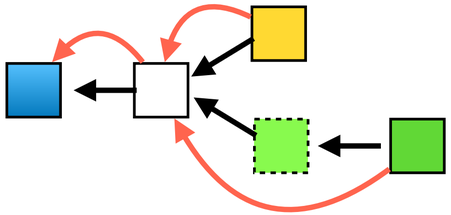
  Set $lock_3$ to □

HotStuff (3 chain)

Example

Example

- 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



Node	$lock_3$
n_0	
n_1	faulty
n_2	
n_3	

HotStuff (3 chain)

Example

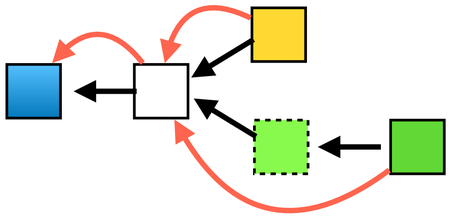
Example

- assume node n_0 is new leader.
- n_0 can extend 🟩, or 🟨
  - both can be signed by n_0, n_2, n_3

New leader

ask $2f + 1$ nodes for last certificate

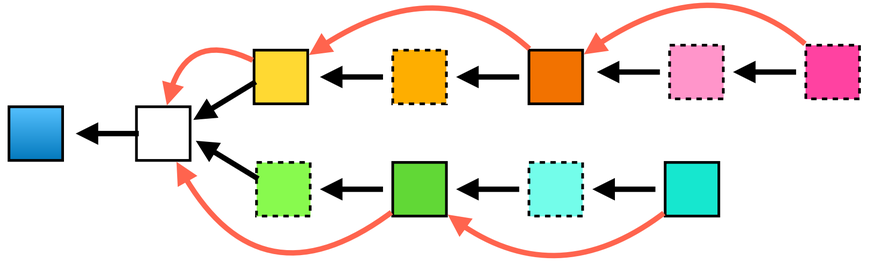
- do not wait for Δ



Node	$lock_3$
n_0	☐
n_1	faulty
n_2	🟦
n_3	☐

HotStuff (3 chain)

Example



Node	$lock_3$	last
n_0	■	■
n_1	faulty	
n_2	□	■
n_3	■	■

Example 2

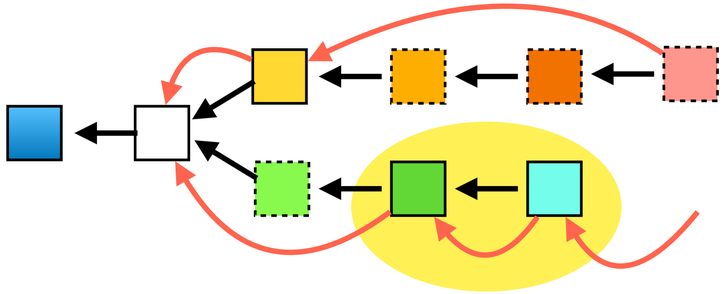
- New block ▢
  can be signed by all correct nodes.
- For n_0, n_2 block extends $lock_3$
- For n_3, ▢ extends ■, which has higher depth than $lock_3$

HotStuff (3 chain)

Confirmation

Example

- New block
  can be signed by n_0, n_2 and faulty n_1
- is not confirmed



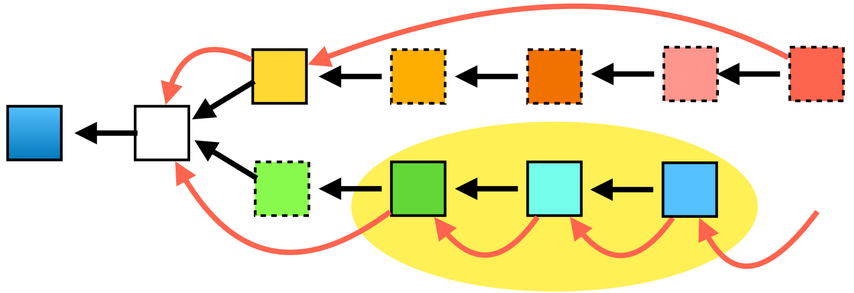
Node	$lock_3$	last
n_0		
n_1	faulty	
n_2		
n_3		

HotStuff (3 chain)

Confirmation

Example

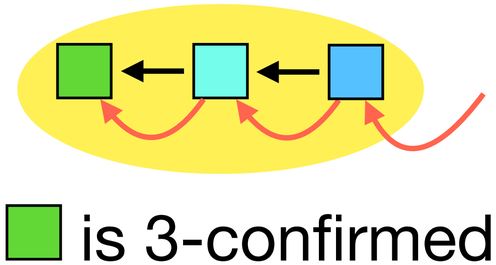
- If n_2, n_3 sign ■, they set $lock_3$ to ■
- ■ is not confirmed



Node	$lock_3$	last
n_0	□	■
n_1	faulty	
n_2	■	■
n_3	■	■

HotStuff (3-chain)

Confirmation



is 3-confirmed

Def.: A block is 3-**confirmed** if the block and it's 2 next 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 $lock_3$ to the confirmed node.

Hybrid blockchains

Confirmation

BFT vs PoW

BFT (HotStuff)

- Confirmation requires 2 - 3 blocks
- Confirmation requires seconds
- Confirmed transactions are secure, as long as failure threshhold holds.

PoW (Bitcoin)

- Confirmation requires 6 blocks
- Confirmation requires 1 hours
- Confirmation is probabilistic

Hybrid blockchain
BFT & PoW

- A **hybrid blockchain** achieves BFT style confirmation in a permissionless system.

ByzCoin: [UsenixSec] like Bitcoin-NG, but require certificate for microblock.

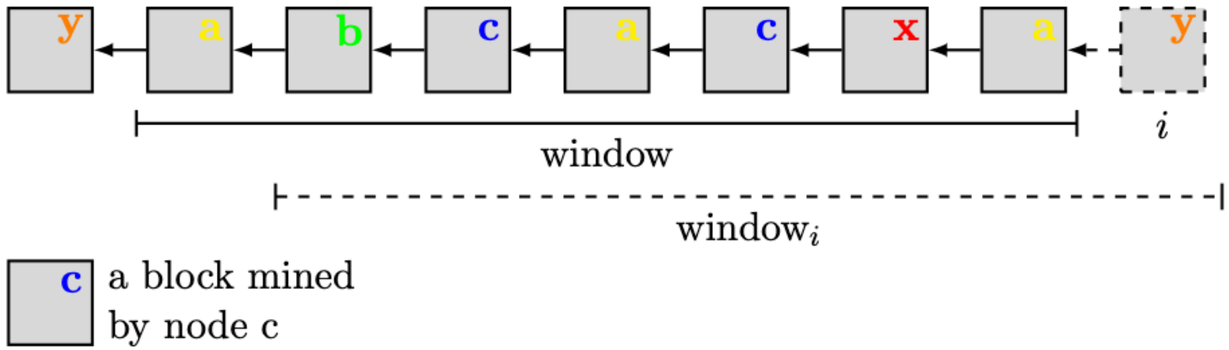
- Who may sign: People finding last n key blocks.
    - Possibly first n out of last $n + \delta$ key blocks (e.g. $\delta = 6$)
- Failure threshold: Assume adversary does not get more then f out of n consecutive key blocks.

Hybrid blockchain

ByzCoin

In ByzCoin, a miner has a *voting share*, based on the number of blocks he has mined in the current window.

Example:



Voting share of a is 3, thus signatures from a and c make a certificate in the current window, but no longer in the next.

Hybrid blockchain

ByzCoin

In ByzCoin, a miner has a *voting share*, based on the number of blocks he has mined in the current window.

Why use a sliding window: Recent nodes are more likely still online!

Hybrid blockchain

Casper FFG, Proof of Stake (PoS) & BFT

Idea: Nodes voting/signing in BFT are those that have frozen money in a transaction in last n blocks.

The voting share of a node is the amount of his deposit, devided by the total frozen money.

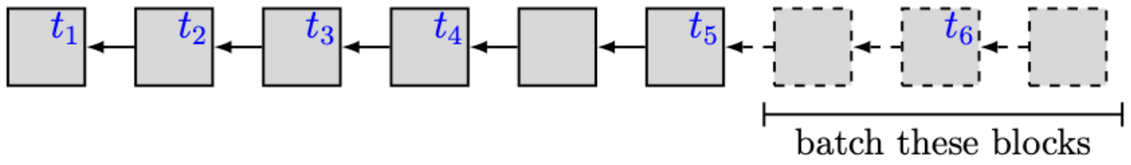
Hybrid blockchain

Casper FFG, Proof of Stake (PoS) & BFT

Idea: Nodes voting/signing in BFT are those that have frozen money in a transaction in last n blocks.

The *voting share* of a node is the amount of his deposit, devided by the total frozen money.

Example:



signatures from a, b and c make a certificate

transaction	deposited stake	submitted by
t_1	2 eth	a
t_2	2 eth	b
t_3	1 eth	c
t_4	1 eth	d
t_5	1 eth	e
total	7 eth	

Hybrid blockchain

Casper FFG, Proof of Stake (PoS) & BFT

Idea: Nodes voting/signing in BFT are those that have frozen money in a transaction in last n blocks.

The *voting share* of a node is the amount of his deposit, devided by the total frozen money.

Leader election: Casper FFG relies on PoW to propose blocks!

Hybrid blockchain
PoS & BFT

Idea: Nodes voting/signing in BFT are those that have frozen money in a transaction in last n blocks.

The *voting share* of a node is the amount of his deposit, devided by the total frozen money.

Leader election: [Algorand] *(simplified)*

- Blocks contain a $nonce$.
- Sign and hash nonce, to see if I can publish a new block.
  $H(Sign_{pk}(nonce_{i-1}) <$ stake $\cdot\, d$?
- Create new nonce: $nonce_i = H(Sign_{pk}(nonce_{i-1})$

Alternative PoW - Proof of Stake (recap)

What is the scarce resource?

- PPCoin (Peercoin)

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d_0 \cdot \text{coin}(addr)$$

Problems:

- **Predictability** (will I get the next block)
- **can PoW** (change transactions to get next block)
- **Non deciding** (can mine on two forks)
- **History rewrite** (can rewrite complete history)

Hybrid blockchain
PoS & BFT

Algorand:

- Use of BFT solves **Non-deciding** and **history rewrite** problem
- Leader election does not allow PoW, since pk public key is fixed
- To reduce predictability:
  - Change difficulty to allow multiple leaders
  - Nodes wait for leaders proposal for Δ time
  - Nodes sign proposal with lowest nonce

Hybrid blockchain

PoS & BFT

Algorand:

- Scaling problem: Collecting signature from all nodes is costly.
- Use same mechanism as in leader election, to select a committee, that is resposible for creating a certificate.

Repetition questions

HotStuff

Question: What is the main advantage of HotStuff over simple HotStuff?

Answer: No need to wait for Δ time on leader change

Repetition questions

ByzCoin

Question: Why combine PoW and BFT?

Answer: Get quick confirmation in permissionless setting.

Question: How are the nodes running a BFT protocol selected in ByzCoin?

Answer: Sliding window over key blocks.

Question: In ByzCoin, how can you avoid diagreement on committee through forks in key-blocks

Answer: Do not include the most recent δ (e.g. 6) key blocks in committee.

Repetition questions

Casper FFG

Question: How do you decide the voting power of a node or private key in Casper FFG?

Answer: Fraction of the nodes frozen stake, devided by all stake.

Repetition questions

Algorand

Question: How does algorand solve the common problems in Proof of Stake?