BFT and Hybrid solutions

HotStuff, ByzCoin, CasperFFG, and Algorand

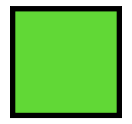
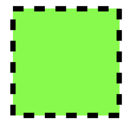
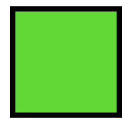
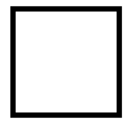
Leander Jehl

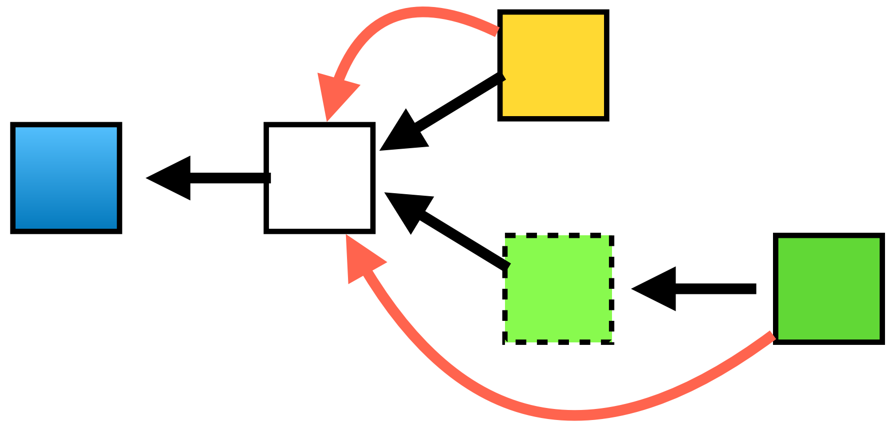
HotStuff BFT

Simple HotStuff
Certificates

- A **certificate** contains signatures from more than 2/3 of the nodes.

Last week: Every block contains a certificate for its parent.
Now: *Every block contains a certificate for some ancestor.*

- *Example:*
  -  has parent 
  -  has certificate for 



Simple HotStuff (2 chain)

Rules

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

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.

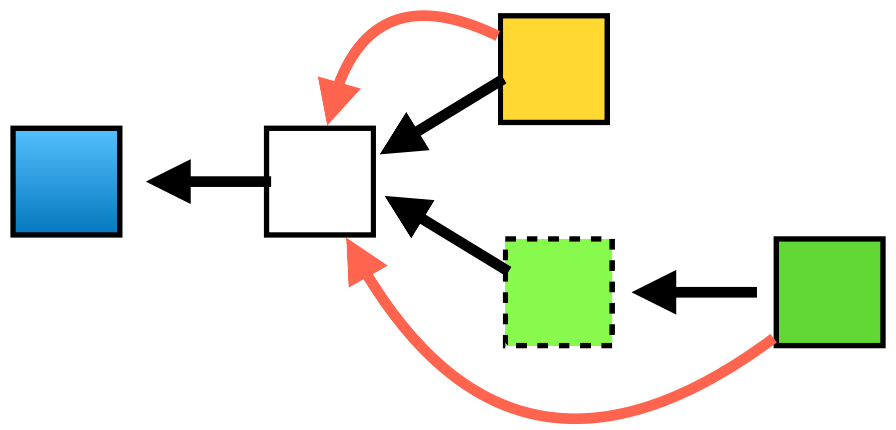
Obs: a node may update the locked block, based on the certificate included in a block.

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

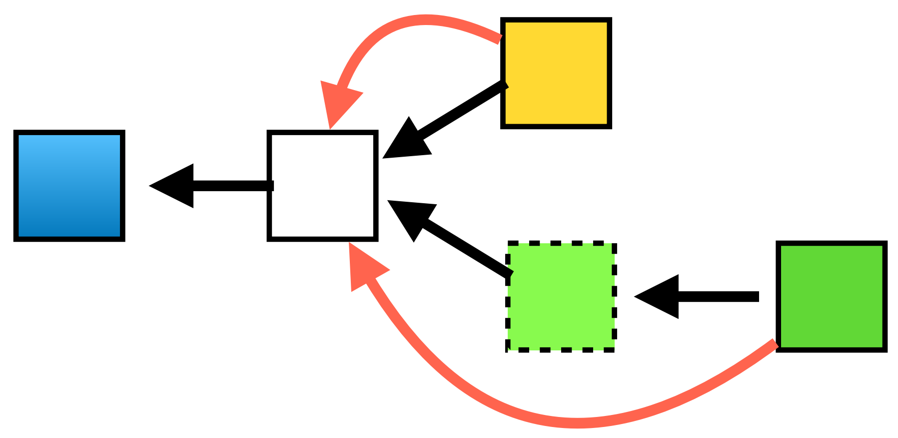
Example

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

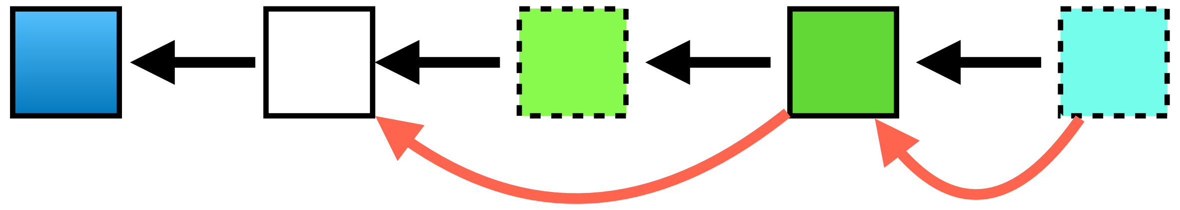
- **Rule 1:** After signing a block as depth d,
  a node may only sign at depth $d' > d$.

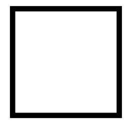
Every node maintains the $lock_3$ **block,** i.e. the block at largest hight, such that this block and one successor have a certificate.

- **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:** 

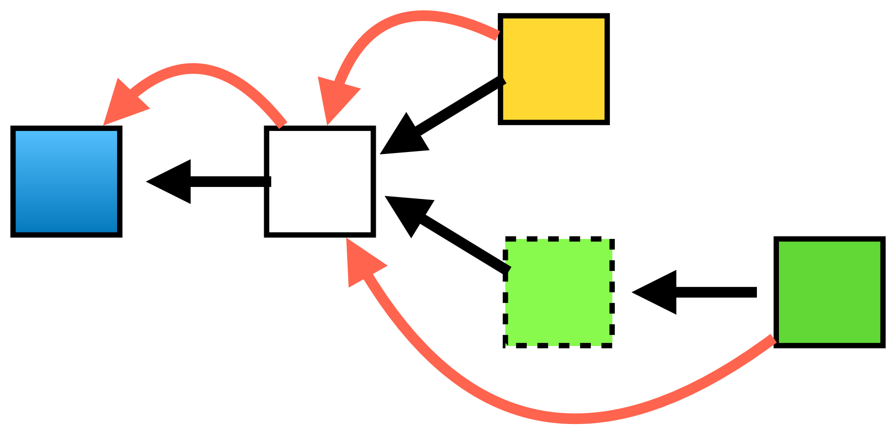
  Set $lock_3$ to 

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

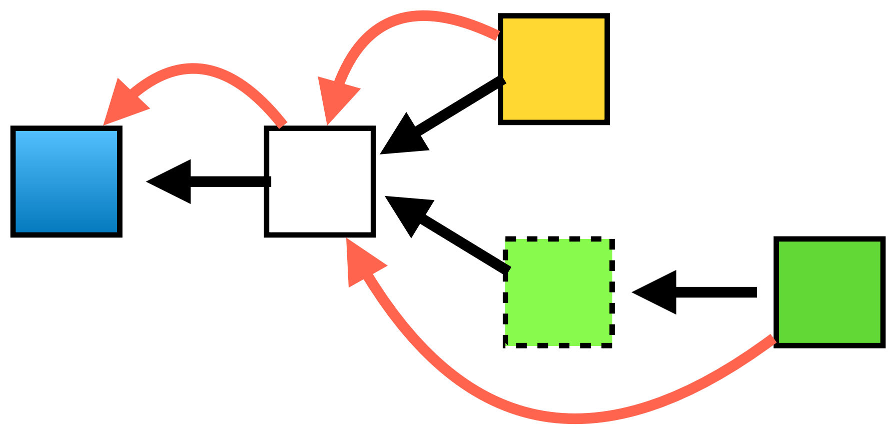
Example

- 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

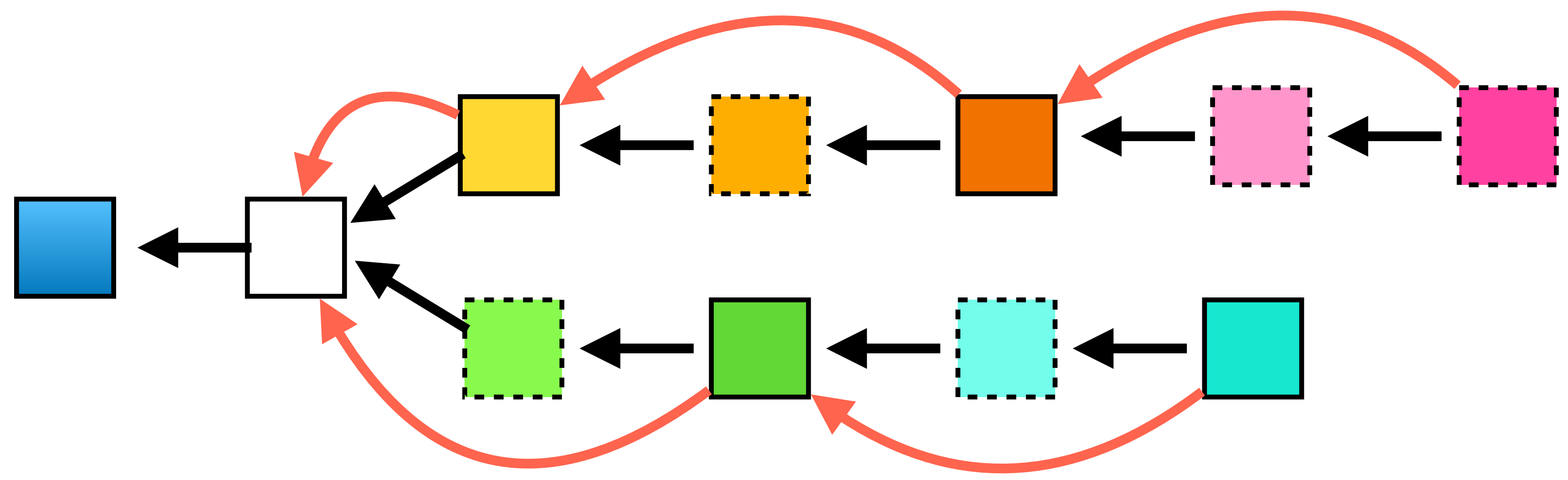


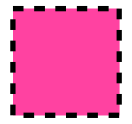
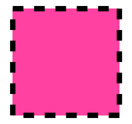
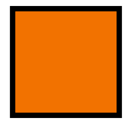
Node	$lock_3$
n_0	
n_1	faulty
n_2	
n_3	

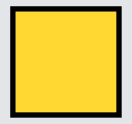
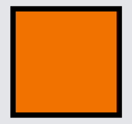
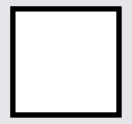
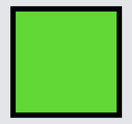
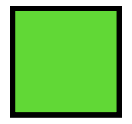
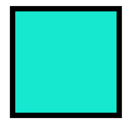
HotStuff (3 chain)

Example

Example 2



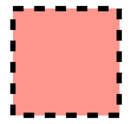
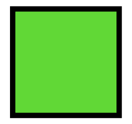
- New block 
  can be signed by all correct nodes.
- For n_0, n_2 block extends $lock_3$
- For n_3,  extends , which has higher depth than $lock_3$

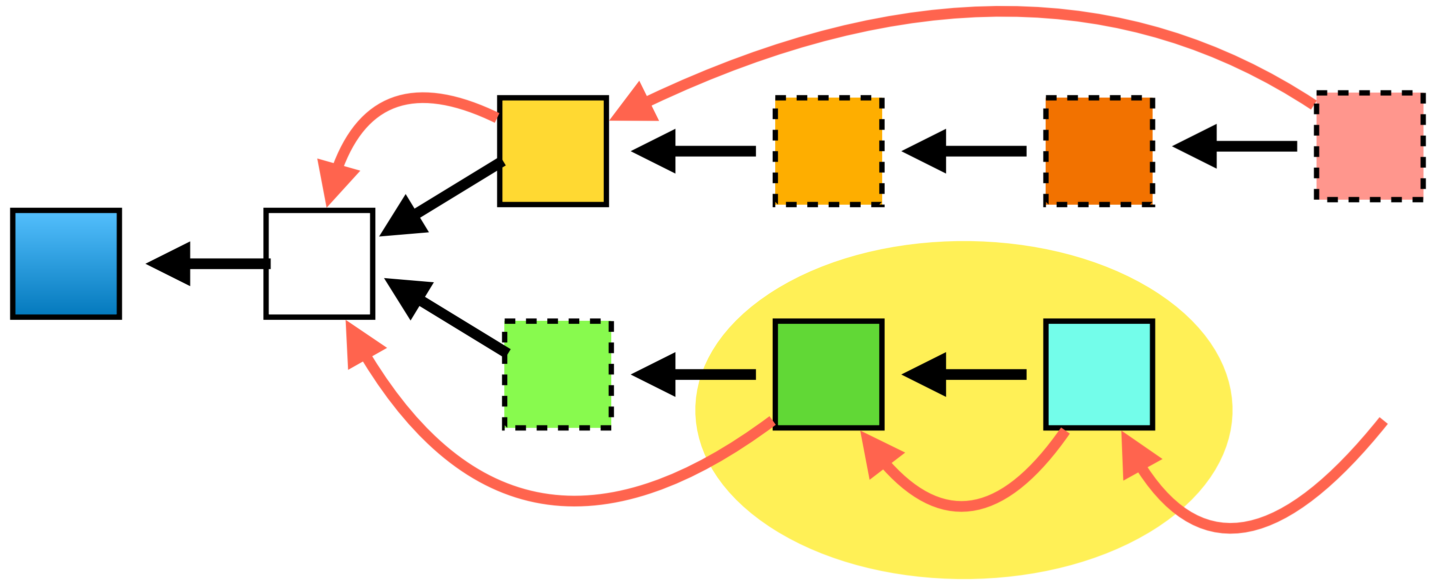
Node	$lock_3$	last
n_0		
n_1	faulty	
n_2		
n_3		

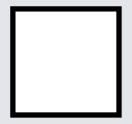
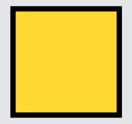
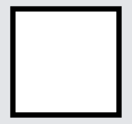
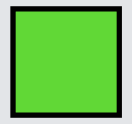
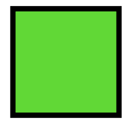
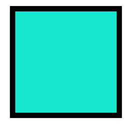
HotStuff (3 chain)

Confirmation

Example

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



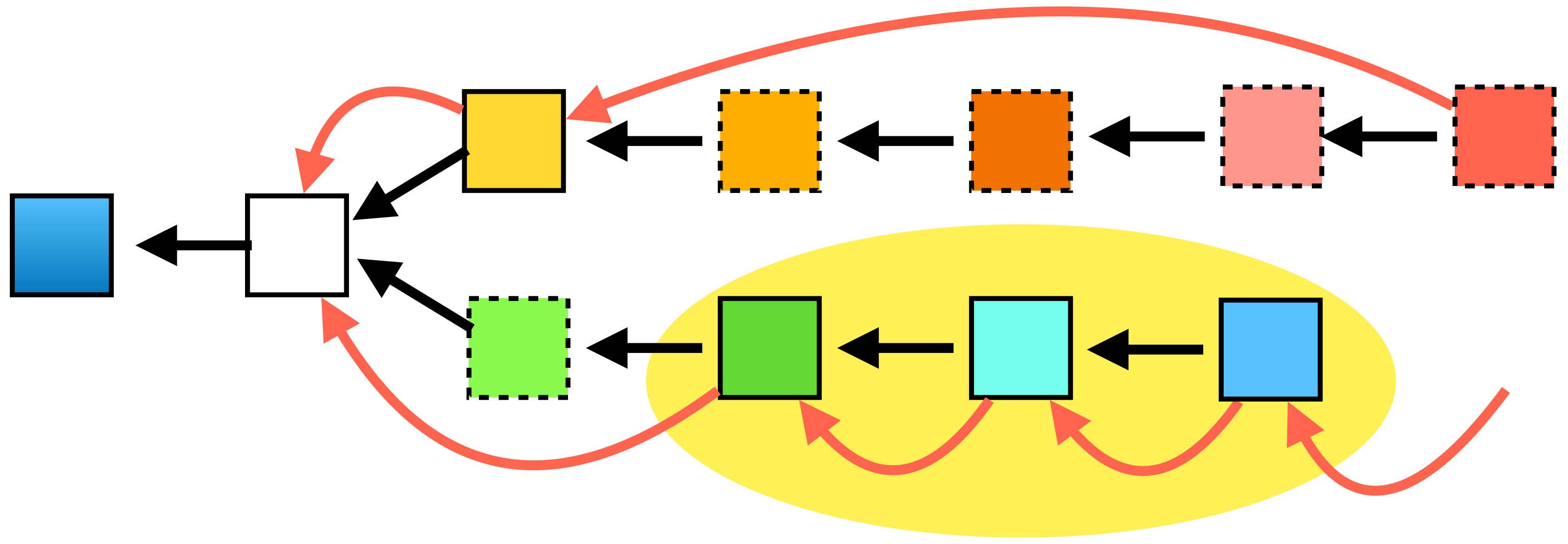
Node	$lock_3$	last
n_0		
n_1	faulty	
n_2		
n_3		

HotStuff (3 chain)

Confirmation

Example

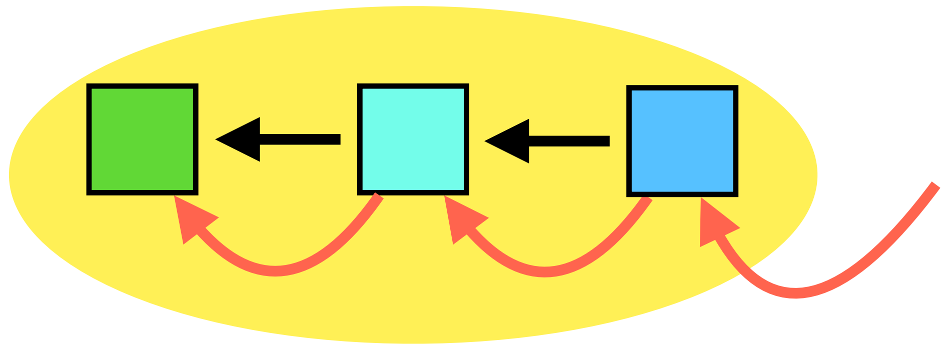
- If n_2, n_3 sign ,
  they set $lock_3$ to
- is not confirmed

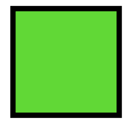


Node	$lock_3$	last
n_0		
n_1	faulty	
n_2		
n_3		

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.

Rational behaviour

Rational nodes

Possible attacks

Can decrease their cost by:

- Not doing work, e.g.

Can increase reward/utility:

Rational nodes

Possible attacks

Can increase their utility by:

- Not doing work, e.g.
  - not verifying block
    (save computation)
  - not signing block
    (save computation & banwidth)
  - turn of node
  - run slow/failure prone node

Can increase reward/utility:

Rational nodes

Possible attacks

Can increase their utility by:

- Not doing work, e.g.
    - not verifying block

reward nodes, who’s signature is included in a certificate

- turn of node
- run slow/failure prone node

Can increase reward/utility:

Rational nodes

Possible attacks

Can increase their utility by:

- Not doing work, e.g.
  - not verifying block

reward nodes, who's signature is included in a certificate

creates new attack

  - turn of node
  - run slow/failure prone node

Can increase reward/utility:

- by signing every possible block

punish nodes that disobey protocol

- by influencing who gets the reward

reward leaders, who include many signatures

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.

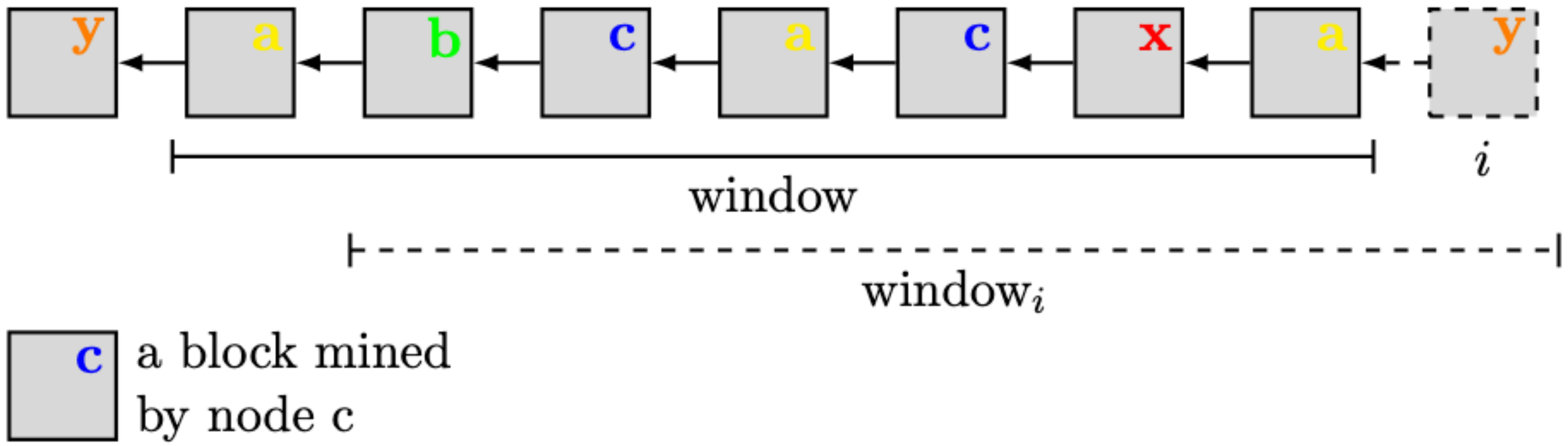
- Who may sign: People finding last n key blocks.
    - Possibly first n out of last $n + \delta$ key blocks (e.g. $\delta = 6$)
- Failure threshold: Assume adversary does not get more then f out of n consecutive key blocks.

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.

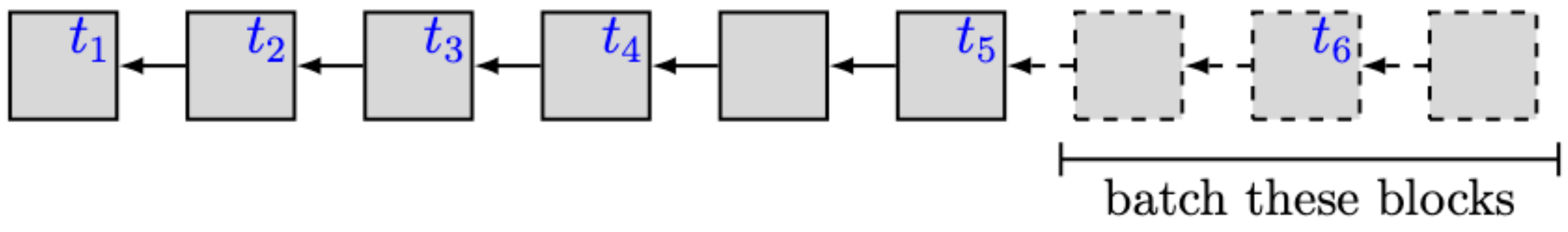
Hybrid blockchain

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: Can be done similar to Peercoin.

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 (**look in the future**)**
- **Nothing at stake (**Can work on 2 forks**)**
- **Possibly unfair (**rich get richer**)**
- **Possible to PoW (**stake grinding**)**
- **History rewrite (**Long range attacks**)**

Hybrid blockchain
PoS & BFT

Algorand:

- Use of BFT solves **history rewrite** problem
- Reward to all signers solves **possible unfair**
- BFT rules solve **nothing at stake** problem
    - punishment if rules are not followed

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.